Gerakoulis, Diakoumis P.

MULTIBEAM SATELLITES: THE MULTIPLE ACCESS AND THE SWITCHING PROBLEMS

City University of New York PH.D. 1985

MULTIBEAM SATELLITES:

THE MULTIPLE ACCESS AND THE SWITCHING PROBLEMS

by

DIAKOUMIS P. GERAKOULIS

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1985

This maniscript has been read and accepted from the Graduate Faculty in Engineering in satisfaction of the dissertation for the degree of Doctor of Philosophy

11/30/84
date

Chairman of Examining Committee

11/30/84
date

Executive Officer

Prof. T. Saadawi
Prof. D. Schilling
Prof. G. Eichmann
Prof. N. Schienberg
Prof. J. Manassah
Supervisory Committee

The City University of New York

ABSTRACT

MULTIBEAM SATELLITES:

THE MULTIPLE ACCESS AND THE SWITHCING PROBLEMS

by

DIAKOUMIS P. GERAKOULIS

Adviser: prof. TAREK SAADAWI

The next generation of communication satellites are expected to employ satellite switched multiple spot beams to increase the system capacity. In this Thesis we study two basic problems encountered in the design of a satellite switched multibeam system, in packed switching enviroment and with "burty" traffic. First is the switching problem from various source to destination spot beams and second is the multiple access problem between the ground stations or users wthin each beam.

The switching problem is studied extensively in chapters 1 and 2. There we assumed the multiple access problem solved on demand basis, forming a set of requests to be switched, called Traffic Matrix T. In chapter 1 we propose and evaluate two scheduling algorithms, which will optimally switch the traffic of the matrix T, in the sense of maximizing transponder utilization. In addition the algorithms have certain other impoved characteristics such

as, both generate small number of switching modes and their overall computational complexity has been reduced. The first algorithm utilizes the idea of Latin Squares, while the in the second one a new approach of the problem has been introduced, the approach of transmitting simultaneously and independently "noncoflicting" submatrices of the traffic matrix.

In chapter 2 we basically, present two methods of minimizing the number of swichings while keeping the transmission time of the traffic matrix as small as possible. The first is the method of superimposing a latin square on the traffic matrix T and transmitting T according to the latin square. This method has been implemented by an efficient and optimum algorithm. A suboptimum algorithm has also been used in this case. The second is the method of "nonconflicting" submatrices used in chapter 1, which is the most appropriate for this type of problem and may be considered as an extention of the first one. Both of the above methods also permit continuous transmission of the message (a number of packets) represented by each entry in the traffic matrix. Simulations have been conducted to examine and compare the transmission time of the traffic matrix.

Within the framework of the multiple access problem we present and analyze a class of Tree Algorithms with variable message length (chapter 3). The Tree Algorithm with variable message length is the one which permits the tranmission of

the user's message, consisting of a number of packets, after the successfull transmission of its first packet. The analysis is presented for small and large number of users. As an application of the above multiple access algorithms to the multibeam satellites we present a hybrid TDMA/Tree scheme, where the switching problem is solved in a fixed manner i.e, the switch follows a preedetermed sequence.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to express my gratitude to my thesis advisor Prof. T.N.Saadawi for his guidance and help throughout the course of this work.

I would like also to thank the members of the commettee and Dean P. Karmel for his help during my Ph.D. studies.

INTRODUCTION

The satellite is an essential part of an integrated computer communications network, it may provide connectivity between two remote points in the system. A communications satellite, placed in geostionary orbit 36,000 km above the earth, receives signals at a frequency f_1 (uplink freq.) amplifies them, translates their frequency to f_2 (downlink freq.) and retransmits them back to earth. The resulting propagation delay of the transmitted signal ranges from 0.25 to 0.27 sec. depending on the location of the earth station. The most commonly used frequency band is the 4/6 GHz for commercial traffic, while the usage of 11/14 GHz band is increasing. The need for using more channel bandwidth is to seek new frequencies (20/30 GHz and 43/45 GHz bands) The rapid growth communication traffic through satellites dictates the need for a mean of sharing the system resources and improving channel utilization. This can be done through frequency reuse which is the main porpose of a multibeam satellite.

The sources of communications traffic are mainly of two types based on their statistical characteristics: a) "stream" traffic such as voice, video etc, and b) "bursty" traffic, such as interactive terminals, transaction systems, etc. Stream traffic is usually better accomodated using circuit switching techniques, while bursty traffic is best accomodated using packet switching. In this thesis we shall

deal exclusively with packet switching i.e., we consider all the traffic in the form of packets (blocks of digital messages 1,000 to 2,000 bits long, and with fixed length). Every packet, in addition to data bits, is assumed to carry a header with the source and destination information, synchronization information etc.

The Multibeam Satellite

A very efficient method in allocating the communication bandwidth provided by the satellite is by using satellite switched multibeam systems. The multibeam satellite has number of spot beam antennas, each covering a separate geographical area or zone, and each zone having a number of ground stations or users. Each beam comprises an uplink and a downlink carrier frequency. On board the satellite is the matrix switch which makes the connections from each source to each destination zone without conflict. Also on board are a number of transponders that translate the uplink carrier to downlink one. Such a system is shown concepually in Fig 1.

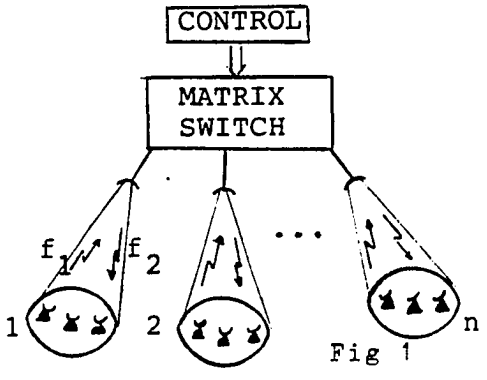


Fig 1

This system offers the great advantage which is to provide frequency reuse and thus more efficient utilization of the

frequency spectrum.

In the design of such a multibeam satellite system we face two basic problems. First is the Multiple Access problem which is the problem of organizing access of all stations in the zone that contending for the same uplink. Second is the switching problem. This is the problem of setting up the satellite switch so that the packets arriving at the uplinks are switched to the appropriate downlink without conflict, which means each time must be one to one connectivity between uplink and downlink.

Satellite Multiple Access

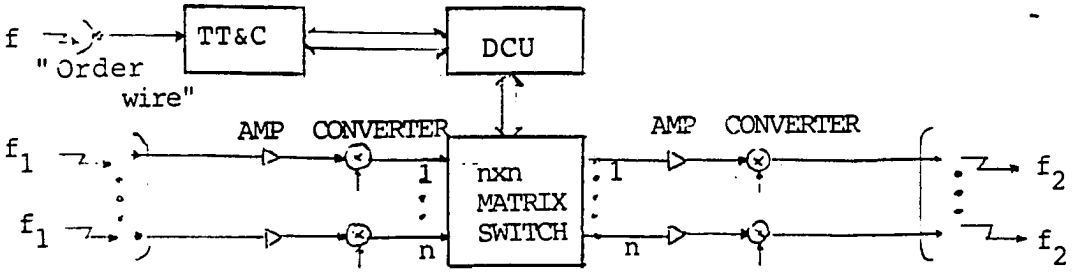
Since the satellite channel share a population of users the problem of multiple access arises. To achieve efficient channel utilization many multiple access protocols have been developed over the past decade. We distiguish two major categories of packet oriented multiple access protocols for bursty traffic a) contention b) reservation. Unlike the fixed access techniques (TDMA, FDMA) in both of the above categories the channel capacity is dynmically allocated to the users. Under contention protocols there is no attempte to coordinate transmission of user's packet and to avoid collision of transmitted packets entirely; while under reservation protocols a reservation channel is provided for ready users to coordinate transmission and avoid collision. Contention protocols include ALOHA, URN scheme Tree type protocols etc, while in the reservation category include the Round-Rubin scheme, Polling schemes, etc. A Class of Tree

Algorithms With Variable Message Length is presented and analyzed in chapter 3.

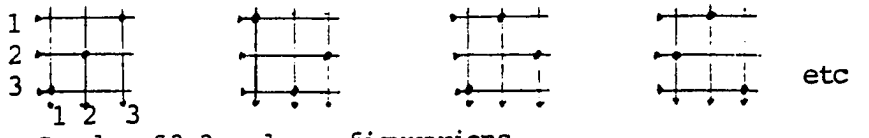
The SS/TDMA concepts

The switching problem we face in the design of a multibeam satellite needs carefull consideration, especially in the case of packet switching, in order to achieve system efficiency. There are two basic approaches to the switching problem. In the first one, the entire frequency band is divided into sub-bands by means of filters and the output of each filter is to be connected to the appropriate downlink beam. This approach is studied in [1], [2] and [3].

In the second approach, called Satellite Switched/ Time Division Multiple Access (SS/TDMA), the entire bandwidth is used for uplink and downlink transmission. The SS/TDMA system incorporates both space division and time division multiple access. A block diagram of an SS/TDMA is shown in Fig 2.



a) Simlified block diagram of an SS/TDMA satellite



b) a Sample of 3x3 mode configurarions

Fig 2

The components in Fig 2 are the matrix switch, the Distribution Control Unit (DCU) and the multibeam antennas. Also illustrated are the RF up-downlink convertes or transponders. The switching is performed at RF. A Microwave Switching Matrix (MSM) provides interconnectivity among uplinks and downlinks. Such a MSM has been designed to operate at 4 GHz center frequency. Signals before and after MSM must be amplified. For future SS/TDMA systems switching at baseband is expected. This will require on board processing of signals (demodulation remodulation) but it will provide full potential capability of digital transmission. Further more switching at baseband can be implemented using relatively simple light weight, low power, high speed switches. The DCU contains the matrix switch scheduling algorithm stored in a programable microprocessor which controls the matrix switch and register its switching modes. Also illustrated is the Telemetry Tracking &Command (TT&C)equipment which receives the traffic requirements and keeps track of the switching modes. Also has the capability of reprogramming the DCU by ground command.

Basically, there are two strategies for scheduling the switch positions; first is the fixed strategy i.e., the switch follows a predetermined sequence, used in Hybrid TDMA-Tree scheme of chapter 3. Second is the demand switch scheduling strategy. In this case each ground station uses the low capacity "order wire" to transmit their requests. On board the satellite DCU make a list of all requests, called

Traffic Matrix $T=[t_{ij}]$, where t_{ij} represents the traffic demand in packets from uplink zone i to downlink zone j. Next the DCU scheduling algorithm is applied to decompose T into a number of Modes. **Mode** is the switch connectivity arrangment at any given instance, which permites transmission of the traffic without conflict. Fig 2b shows some 3x3 mode configurations. The schedulling decisions are transmitted back to the earth stations and each station transmit according to the schedule. Global synchronization is assumed. In chapters 1 and 2 we investigate the problem of schedulling algorithms for SS/TDMA systems. The algorithms we propose have main objective to make the system as efficient as possible.

Related reserch on SS/TDMA

The research on the scheduling problem for SS/TDMA systems started few years ago and is still going on. Related work can be found in [4] where Gadre and Stern compares a number of possible configurations and protocols for SS/TDMA. A basic scheduling algorithm is introduced in [13] by Inukai. A generalization of the SS/TDMA system and the problem of scheduling the traffic is intoduced in [16] and [18]. There the number of transponders can be less or equal to the number of zones or the beams may have variable bandwidth. In [17] Gopal et al attempted to optimize the system on the basis of minimizing the packet waitting time. Another variation of the problem is examined in [10] they consider the zones partially overlapping. Integrating packet and

circuit switching is considered in [9], where Frank and Stern simulated several algorithms. The problem of minimizing the number of switching modes is examined in [21] and [22] by Gopal, Natarajan et al. Finally an introductory presentation to the SS/TDMA system with scanning beams is given in [7] and [8] by Acompara et al.

Develoment and Technology of SS/TDMA systems

Several SS/TDMA spacecrafts are currently under consideration (see [6]); among them include the INTELSAT VI satellite, scheduled for launch in 1986, it will operate a 6x6 dynamic switch which will provide full interconnectivity between two hemisphere beams and 4 overlaid zone beams, two in each hemisphere, at 4/6 GHz carriers. Also it will incorporate a 8x8 static switch which will maintain a constant configuration for relatively long periods at 14/11 Ghz. Another system planned for launch in mid-1988 is the NASAs Advaced Communication Technology Satellite (ACTAS) which will operate in the 30/20 GHz band with 750 MHz bandwidth channels using 3x3 IF matrix switch and also scanning beams. A third system planned is the Japanese 30/20 GHz SS/TDMA satellite with 4x4 microwave IF matrix switch which will serve japanese mainland divided into 4 disjoint zones.

The key component on the focus of SS/TDMA technology is the satellite matrix switch. The microwave switch device technology is shown in Table I.

Switching Technology	Switching Speed	Device Insertion Gain	DC Power Requirements	Relative Size and Weight
Ferrite	0.1 ~ 1 μs	−.25 dB	high current (1A)	large, heavy
Pin Diode	10 − 100 ns	−.25 dB	low voltage low current (10 ma)	small
Single Gate MESFET	0.1 ~ 1 ns	−3 dB	low voltage (3V) low current (10 ma)	small
Dual Gate MESFET	0.1 ~ 1 ns	+15 dB	low voltage (3V) low current (30 ma)	small

Table 1

One device that stands out is the +15 db insertion gain dual gate FETs. Under contract with NASA, Ford Aerospace and GE have studied several matrix architectures and concluned that the coupler crossbarr matrix switch will best suit future SS/TDMA requirements. GE has developed a 5x5 matrix switch with 15 ns switching speed and 20 GHz bandwidth.

Assumptions in the formulation of the problem

In the investigation of the multiple access and the switching problems for multibeam satellites, considered in this thesis, we have made the basic assumption that all the traffic is in the form of fixed length packets and that the channel time is slotted. In chapters 1 and 2 we have considered the optimization problem of scheduling the traffic in an SS/TDMA system in the sense of maximizing transponeder utilization or equivalently minimizing the schedule length. This problem is slightly different from the problem of global minimization of the packet delay i.e., end to end optimization, which is considered in [17] and also in chapter 3 of this thesis. Thus the packet transmission sequense is not necessarily the same as the packet arrival sequense to the users buffer.

In chapter 2 we considered the additional constrain of minimizing the number of switching modes. While in both optimization problems (chapters 1 and 2) we try to meet our objectives with algorithms of minimum complexity.

In chapter 3 we considered a class of Tree Algorithms with variable message length with respect to the multiple access problem and the Hybrid TDMA-Tree algorithm as another formulation of the multibeam accessing problem.

CHAPTER 1

1. IMPROVED SCHEDULING ALGORITHMS FOR SS/TDMA SYSTEMS

In this chapter we deal with the switch scheduling problem for SS/TDMA systems where special emphasis is given to the efficiency of the algorithm. More specifically this problem has as follows: given a Traffic Matrix $T=[t_{ij}]$ find a scheduling algorithm which routes the traffic of T from various source to various destination zones without conflict. We assume that the uplink beams are equal to the downlink beams equal to the number of transponders.

Scheduling algorithm is the procedure of decomposing a traffic matrix T into a sum of mode matrices i.e., $T=T_1+T_2+\ldots+T_L$. **Mode Matrix** T_i is a matrix with at most one entry on each row and each column. A mode matrix may be considered as the adjacency matrix of a matched (weighted) bipartite graph. As shown in Fig 1.1, a matched bipartite graph is the appropriate switching mode configuration that permits one to one connection between source and destination zones and thus no conflict occurs during the transmission of packets.

$$T_i = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

uplink ↓

downlink → 1 2 3 4

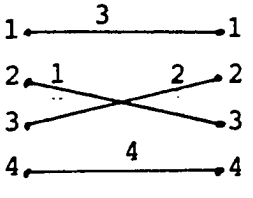


Fig 1.1

Such a switching mode lasts as many slots or packets as

the largest entry in the mode matrix T_i, which we call **Mode Duration** denoted by $|T_i|$.

The total duration $\sum_{i=1}^{k} |T_i|$, needed to schedule the total traffic of T is called **Schedule Length** and is equal to the TDMA frame.

In order to have an efficient algorithm we need to minimize the schedule length or equivalently maximize transponder utilization (throughput). Also at the same time the number of mode matrices must be kept as small as possible in order to reduce the overhead due to switching. The total number of possible nxn mode matrices is of the order of n!, and an algorithm that which tries all the possible combinations until it finds the right one would be "horrible". Thus, in addition, the efficient algorithm is the one that gives the right answer in polynomial time and of order as small as possible. The following table gives us a feeling about the differences in growth rates among several typical complexity functions. we consider each step performed in 1 sec.

f(n) \ n	10	30	60
n^3	.001sec	.027sec	.216sec
n^5	.1sec	24.3sec	13 min
2^n	.001sec	17.9min	366centuries

Table 1.

The above type of scheduling algorithms i.e., decomposing the traffic matrix into mode matrices, is the common way of scheduling the traffic used in [11]-[19], also used in the

Scheduling Algorithm 1 of this chapter. An illustration of this method is given in the first case of the following example.

A new approach of solving the scheduling problem is introduced and used in the Scheduling Algorithm 2 of this chapter. This is the approach of "nonconflicting" submatrices or blocks. Nonconflicting submatrices are two or more submatrices with no common rows or columns. Thus each mode matrix of the first method may be replaced by a "switching matrix" consisting of a number of sets of nonconflicting blocks where each block in a set can be transmitted simultaneusly and independently from each other. This approach is illustrated in the second case of the following example.

Both proposed algorithms in this chapter are optimal, in the sense of minimizing the schedule length. Also both of them have the number of switchings much lower than its upper bound (n^2-2n+2, n:number of zones). In addition an impovement has been made on the reduction of the computational complexity of the proposed algorithms, compared with the existing onces, ref. [11]-[19].

First Case

$T =$

	A	B	C	D
A	2	2	1	2
B	3	2	2	2
C	1	3	4	1
D	2	1	1	5

← Destination Zones

↑ Source Zones

$$T = \begin{bmatrix} & 2 & & \\ 3 & & & \\ & & 4 & \\ & & & 5 \end{bmatrix} + \begin{bmatrix} & & 2 & \\ & 2 & & \\ & 3 & & \\ 2 & & & \end{bmatrix} + \begin{bmatrix} 2 & & & \\ & 2 & & \\ & & 1 & \\ & 1 & & \end{bmatrix} + \begin{bmatrix} & 1 & & \\ & & & 2 \\ 1 & & & \\ & 1 & & \end{bmatrix}$$

From A	toB	toD	to A	C
B	toA	toC	to B	toD
C	to C	toB	D	A
D	to D	toA	C	B

5 — 3 — 2 — 2 — Mode durations

Schedule length = 5+3+2+2=12

Second Case

$$T = \left[\begin{array}{cc:cc} 2 & 2 & & \\ 3 & 2 & & \\ \hdashline & & 4 & 1 \\ & & 1 & 5 \end{array}\right] + \left[\begin{array}{cc:cc} & & 1 & 2 \\ & & 2 & 2 \\ \hdashline 1 & 3 & & \\ 2 & 1 & & \end{array}\right]$$

From A	toA	toB	C	toD
B	toB	to A	toC	toC
C	to C	D	to B	A
D	to D	C	to A	B

6 — 4

Schedule length = 6+4=10

2. MATHEMATICAL FOUNDATION OF THE PROBLEM

Definitions: Traffic Matrix $T=[t_{ij}]$ is an nxn matrix which has non-negative integer entries. We call any row or column of T a **Line** and the **Line Sum** is the sum of entries in a line. The maximum line sum, $S=\max\{r_1,r_2,\ldots,r_n;c_1,c_2,\ldots,c_n\}$ where r_i is the line sum of row i and c_j the line sum of culumn j, is aslo called **Critical Line Sum** and any line with line sum equal S is said to be **Critical Line**

An nxn matrix which has all line sums equal is called **Quasi-Doubly Stochastic** (QDS) matrix.

Theorem1: For any given nxn traffic matrix T with critical line sum equal S, it is always possible to add nonegati. integers to the entries of T so that to obtain an nxn QDS matrix with line sum equal to S.

Proof: For any entry t_{ij} of T we add the quantity $q=\min\{S-r_i,S-c_j\}$ updating meanwhile r_i and c_j until all line sums become equal to S. This is possible because if we assume $r_i<S$ and no elements to the row i can be added because every $c_j=S$, then $\sum_{j=1}^{n}\sum_{i=1}^{n}t_{ij}=\sum_{i=1}^{n}r_i<\sum_{j=1}^{n}c_j=\sum_{i=1}^{n}\sum_{j=1}^{n}t_{ij}=nS$ which is a contradiction.

The nonnegative integers added to each entry t_{ij} is referred to as **dummy traffic.**

Definitions: 1) A set of n nonzero entries of the traffic matrix T, say $(x_1,x_2,\ldots x_n)$, is called **System of Distinct Representatives** (SDR) if they are distinct and there is at most one such entry in each row and column.

2) A **Permutation Matrix** is a matrix which has a single entry 1 in each line and all other entries zero. A permutation matrix $P=[p_{ij}]$ is said to **Cover** the ijth entry if $p_{ij}=1$. Two permutation matrices are said to be **Intersecting** at the entry p_{ij} if both cover the entry p_{ij}. If two permutation matrices have no entry p_{ij} that intersects are said to be **Nonintersecting**

3) **Latin Square** (LS) based on a set of n elements $\underline{g}=\{a_1,a_2,\ldots,a_n\}$ is an nxn matrix where the n rows are n-permutations of the elements of $\underline{g}$ and where each element occurs exactly once in each row and each column.

For example if $\underline{g}=\{a,b,c,d\}$ an 4x4 latin square is the fowllowing:

$$L=\begin{bmatrix} c & d & b & a \\ a & c & d & b \\ b & a & c & d \\ d & b & a & c \end{bmatrix}$$

Theorem 2: Let D be an nxn QDS matrix with line sum equal S, then $D=\sum_{i=1}^{n} c_i P_i$ where P_i`s are the permutation matrices and $c_j$`s are positive reals such that $\sum_{i=1}^{n} c_i=S$

Proof: This theorem is proved in [13] for the normalized case S=1

Theorem 3: There exist at least n!(n-1)!...2!1! nxn latin squares

(proof is given in [23])

Theorem 4: If the matrix L is a LS based on the set $\underline{g}=\{a_1,a_2,\ldots a_n\}$ i.e. the entries of L are $a_1,a_2,\ldots,a_n$ with

no repeated entries on, any line, then $L = \sum_{i=1}^{n} a_i P_i$ where P_i are nonintersecting permutation matrices

Proof: This comes directly from the definition of the LS since each element occurs exactly once on each line of L, permutation matrices form nonintersecting SDRs

Theorem 5: The minimum possiple schedule length that can be achieved by any scheduling algorithm is equal to the maximum line sum S, i.e. $\sum_{i=1}^{L} |T_i| \geq S$

Proof: If we decompose the traffic matrix T into L mode matrices T_i where in each one the element belonging to the critical line is greater or equal to the other elements, then $\sum_{i=1}^{L} |T_i| = S$ and this is the best possible.

The scheduling algoritms that always achieve the minimum schedule length S is said to be **Optimal**. Optimal scheduling algorithms always exist this is guaranted by theorems 1 and 2.

3. SCHEDULING ALGORITHM 1 (SA1)

STEP 1: Given an nxn traffic matrix T. If T is not QDS with line sum S add dummy traffic to it, as indicated by theorem 1, to form one which is QDS with line sum S; lets call it D. Initialize by setting $h \leftarrow 1$, $D_1 \leftarrow D$

STEP 2: If D_h has no zero entries, go to step 3.

If D_h has zero entries but the maximum number of zero entries on every line, m, is less than n/2 i.e. $0 < m < n/2$, go to step 4.

If D_h has at least one line where the number of zeros is greater than n/2 i.e $n/2 \leq m < n$, go to step 5.

STEP 3: Apply the Max-Min algorithm (see Appendix I) to find an SDR, say $\{x_{1_k}, x_{2_k}, \ldots, x_{n_k}\}$. Form a mode matrix $a_k P_k$ where $a_k = \{x_{1_k}, x_{2_k}, \ldots x_{n_k}\}$ and P_k the corresponding permutation matrix to the SDR.

Construct any latin square which has P_k as one of its nonintersecting permutation matrices. Based on the above latin square we extract the other (n-1) nonintersecting permutation matrices, (P_i i=1,n and $i \neq k$), each corresponding to an SDR in D_h, say $\{x_{1_i}, x_{2_i}, \ldots, x_{n_i}\}$. Now we pick the minimum element in each SDR $a_i = \min\{x_{1_i}, x_{2_i}, \ldots x_{n_i}\}$ for i=1,..,n; and we form a matrix L_h which is the sum of the n mode matrices $a_i P_i$ found above i.e. $L_h = \sum_{i=1}^{n} a_i P_i$ with all $a_i > 0$

Set $h \leftarrow h+1$ and $D_h \leftarrow D_h - L_h$. If $D_h \neq 0$ go to step 2. If $D_h = 0$ go to step 6.

STEP 4: Find m nonintersecting SDRs to cover all the zero entries of D_h.

Choose among the nonzero elements in the SDR the smallest possible. (The above can be done by forming a matrix $D_h = d_{max} - D_h$; where d_{max} is the maximum element in D_h, and appling Max-Min algorithm m times). Form any latin square which has the m nonintersecting permutation matrices corresponding to the m SDRs found above. Extract (n-m) SDRs from D_h corresponding to the (n-m) permutation matrices in the latin square which have no zero elements. Thus the minimum element in each of those (n-m) SDRs is $a_i \neq 0$; $a_i = \min\{x_{1_i}, x_{2_i}, \ldots, x_{n_i}\}$ $i=1,\ldots,n-m$. Then we form a matrix $L_h = \sum_{l=1}^{n} a_l P_l$ which has (n-m) mode martices. (m of the a_l's are zero).

Set $h \leftarrow h+1$ and $D_h \leftarrow D_h - L_h$. If $D_h \neq 0$ go to step 2, if $D_h = 0$ go to step 6.

STEP 5: Apply the Max-Min algorithm to find one SDR $\{x_1, x_2, \ldots, x_n\}$ with P_h the corresponding permutation matrix. Take $a_h = \min\{x_1, x_2, \ldots, x_n\}$ and form a matrix $L_h = a_h P_h$.

set $h \leftarrow h+1$ and $D_h \leftarrow D_h - L_h$. If $D_h \neq 0$ repeate step 5, if $D_h = 0$ go to step 6.

STEP 6: The original matrix D can be written as $D = \sum_h L_h$. Substract the dummy traffic from each L_h and transmit its mode matrices.

Two illustrative examples of SA1 are given below.

Example 1.

Assume the traffic matrix is $T=\begin{bmatrix} 2 & 4 & 3 & 1 \\ 5 & 2 & 2 & 2 \\ 1 & 1 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$

STEP 1

$$D=\begin{bmatrix} 2 & (6) & 3 & 1 \\ (5) & 2 & 3 & 2 \\ 1 & 1 & (4) & 6 \\ 4 & 3 & 2 & (3) \end{bmatrix}$$

Line Sum S=12

SDR {6,5,4,3,}

Set $D \longrightarrow D_1$

STEPS 2,3 A L.S. corresponding to the above SDR is given below

$$LS=\begin{bmatrix} d & a & b & c \\ a & b & c & d \\ c & d & a & b \\ b & c & d & a \end{bmatrix}$$

a=min{6,5,4,3}

b=min{3,2,6,4}

c=min{1,3,1,3}

d=min{2,2,1,2}

$$L_1=\begin{bmatrix} 1 & 3 & 2 & 1 \\ 3 & 2 & 1 & 1 \\ 1 & 1 & 3 & 2 \\ 2 & 1 & 1 & 3 \end{bmatrix}$$

STEPS 2,5

$$D_2=D_1-L_1=\begin{bmatrix} 1 & 3 & (1) & 0 \\ (2) & 0 & 2 & 1 \\ 0 & 0 & 1 & (4) \\ 2 & (2) & 1 & 0 \end{bmatrix}$$

m=2=n/2

SDR {1,2,4,2}

$$LS=\begin{bmatrix} a & b & c & d \\ c & d & a & b \\ d & a & b & c \\ b & c & d & a \end{bmatrix}$$

a=0

b=min{3,1,1,2}=1

c=min{1,2,4,2}=1

d=0

$$L_2=\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

STEP 5 $D_3=D_1-L_1-L_2$

$$D_3=\begin{bmatrix}1 & (2) & 0 & 0\\ 1 & 0 & (2) & 0\\ 0 & 0 & 0 & (3)\\ (1) & 1 & 1 & 0\end{bmatrix} \qquad L_3=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\end{bmatrix}$$

$$L_4=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1 & 0 & 0\end{bmatrix} \qquad L_5=\begin{bmatrix}0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

Schedule length = $\sum_{i=1}^{5}|L_i|=12=S$

Example 2

$$T=D=\begin{bmatrix}5 & 5 & (0) & 2\\ 3 & 6 & 1 & (2)\\ (0) & 1 & 6 & 5\\ 4 & (0) & 5 & 3\end{bmatrix}$$

Line Sum S=12

$m=1 < n/2$

STEPS 2,4 The LS corresponding to the above SDR covering the zeros is given below

$$LS=\begin{bmatrix}c & d & a & b\\ b & c & d & a\\ a & b & c & d\\ d & a & b & c\end{bmatrix}$$

$a=0 \rightarrow ( )$

$b=\min\{2,3,1,5\}=1$

$c=\min\{5,6,6,3\}=3$

$d=\min\{5,1,5,4\}=1$

$$L_1=\begin{bmatrix}3 & 1 & 0 & 1\\ 1 & 3 & 1 & 0\\ 0 & 1 & 3 & 1\\ 1 & 0 & 1 & 3\end{bmatrix} \qquad D_2=D_1-L_1=\begin{bmatrix}2 & (4) & 0 & 1\\ (2) & 3 & 0 & 2\\ 0 & 0 & 3 & (4)\\ 3 & 0 & (4) & 0\end{bmatrix}$$

STEP 5

$$LS=\begin{bmatrix} a & b & c & d \\ b & d & a & c \\ c & a & d & b \\ d & c & b & a \end{bmatrix}$$

$a=0$

$b=\min\{4,2,4,4\}=2$

$c=0$

$d=\min\{1,3,3,3\}=1$

$$L_2=\begin{bmatrix} 0 & 2 & 0 & 1 \\ 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 0 \end{bmatrix}$$

$$D_3=D_1-L_1-L_2$$

$$D_3=\begin{bmatrix} 2 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 2 & 0 & 2 & 0 \end{bmatrix}=L_3$$ Schedule length=5+3+2+2=12=S

Comment

A partial or incomplete latin square (see [27]) which has all the elements $a_1,\ldots,a_m$ $(1\leq m<n)$ and only these, cannot always be completed, especially when $n/2<m<n$. Completion of the latin square is always possible when $m=1$ and 90% of the time when $m<n/2$.

In SA1 step 4 we need to complete a latin square for $m<n/2$. If in this case completion becomes impossible, then we may reduce m by one and try again. In [27] a simple algorithm is given of completing partial latin squares (pages 469-471).

3.1 EVALUATION OF THE PROPOSED SA1

Theorem 6: The Scheduling Algorithm 1 is an optimal algorithm

Proof: At each iteration the SA1 generates a matrix $L_h = \sum_{i=1}^{n} a_{i_h} P_{i_h}$ (1) with at least one $a_i \neq 0$, and since L_h is a QDS matrix, $D - \sum_{h=1}^{H} L_h$ is also QDS matrix because D is QDS. Thus D can be expressed as $D = \sum_{h=1}^{W} L_h$, $1 < H \leq W$. Using (1) we can write $D = \sum_{h=1}^{W} (\sum_{i=1}^{n} a_{i_h} P_{i_h})$ and according to theorem 2 we have $\sum_{h=1}^{W} \sum_{i=1}^{n} a_{i_h} = S$ i.e. the sum of all nonzero equals to the critical line sum S, which means the SA1 always achieves the minimum i.e. the algorith is optimal.

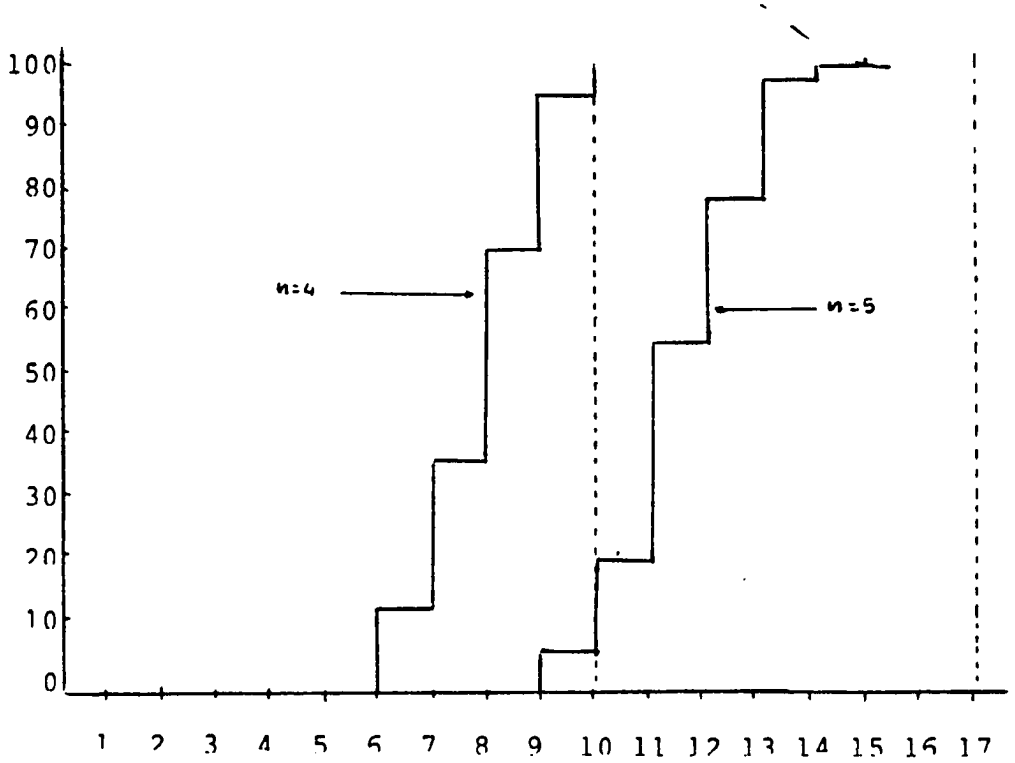
Theorem 7: The number of mode matrices for SA1 is bounded by $n^2 - 2n + 2$ for an nxn traffic matrix, the above bound is the best possible.

Proof: D is expressed as a summation of integer multiples of permutation matrices, $D = \sum_{h=1}^{W} \sum_{i=1}^{n} a_{i_h} P_{i_h}$ where all P_{i_h} for which $a_{i_h} \neq 0$ are distinct (not identical). This is true since all the matrices L_h are distinct, and that because are "guided" by differed SDRs. Given the above consideraion, is proven in [13] that the upper bound on the mode matrices is $n^2 - 2n + 2$. Thus the number of mode matrices for SA1 cannot exceed $n^2 - 2n + 2$.

Simulation results for SA1

To verify theorem 7 the SA1 has been simulated. The entries of the traffic matrices are generated by a uniform integer random number generator, distributed between 0 and 20 .

Fig.1 shows the distribution curves of the number of mode matrices for 100 traffic matrices, and for n=4,5. In both cases the number of mode matrices don't exceed the theoritical upper bound which is 10 for for n=4 and 17 for n=5; in fact for n=4 70% of the traffic matrices have 8 mode matrices,while for n=5 53% have only 11 mode matrices.



Distribution curves of the number of required mode matrices for the SA1,and for n=4,5.

Fig. 3.1

Computational complexity of SA1

The comlexity of SA1 depends on the complexity of the Max-Min algorithm used to exctract a SDR, which is of polynomial order $O(n^3)$ (see Apendix). The Max-Min is applied k times to the traffic matrix T, generating a total complexity of the order $O(kn^3)$;where k is the number of times the max-min algorithm is applied to transmit a traffic matrix T. However for SA1 k is much smaller than the number of mode matrices, which is bounded by n^2-2n+2. This is achieved with the aid of a latin square which permits us to generate many more mode matrices than the number of times the max-min algorithm is applied. Thus the over all complexity is reduced for the SA1 compared with the existing algorithms that have total complexity bounded by $O(n^5)$.(This comes from the fact that the SDR algorithm is applied as many times as the number of mode matrices i.e. $O(n^2)$).

In the next section we propose the Scheduling Algorithm 2 (SA2). The SA2,which generalizes SA1,improves certain characteristing of SA1,permiting more traffic to be transmitted at the beginning of the frame.

4. SCHEDULING ALGORITHM 2 (SA2)

The scheduling algorithm 2 is an extension of the SA1. the fundamendal idea is to divide the traffic matrix T into submatrices named **blocks**. The blocks that have no common lines in T are called **nonconflicting** blocks. Nonconflicting blocks can be considered as independent traffic matrices and thus can switch traffic independently and simultanuesly; more specifically the algorithm has as follows:

STEP 1: Given the nxn traffic matrix T ; if n is a prime number apply the SA1 to transmit its traffic; if not go to step 2.

STEP 2: If T is not a QDS matrix with line sum S add dummy traffic to it to get one which is QDS with line sum equal S, let call it D.

STEP 3: Write n as any product $n=k \cdot m$ where m is a prime number and k is an integer. Apply the Max-Min algorithm to get an SDR. Based on this SDR form k mxm nonconflicting blocks by taking any m elements of the SDR and forming a block(i.e. the m elements are an SDR in the block).

Now based on the set of k noncoflicting blocks formed above, we form (k-1) more sets, of k nonconflicting blocks each.This is done with the aid of a latin square i.e. we form a latin square where the set of k nonconflicting blocks corresponds to an SDR.

STEP 4: Substract traffic so that each nonconflicting block in the set of k is a latin square and all k blocks have equal line sums. We arrange as many zeros as possible belong to the same DSR in each block.

Trans: mit each nonconflicting block in the set of k , independently and simultaneusly with each other, each set of nonflicting blocks after the other.

If the difference, D minus the traffic transimitted at each iteration is not zero go to step 3; otherwise stop.

Tree illustative examples are given below:

Example1

$$T=D=\begin{bmatrix} 2 & (6) & 3 & 1 \\ (5) & 2 & 3 & 2 \\ 1 & 1 & (4) & 2 \\ 4 & 3 & 2 & (3) \end{bmatrix} \qquad S=12$$

STEP 3: $n=4=2\cdot 2$ $k=2$ $m=2$

The LS corresponding to the two sets of nonconflicting blocks is the following

$$LS=\begin{bmatrix} a & b \\ b & a \end{bmatrix}$$

STEP4

1rst Iteration :

$$D_1=\begin{bmatrix} 0 & 5 & & \\ 5 & 0 & & \\ & & 3 & 2 \\ & & 2 & 3 \end{bmatrix} \quad \begin{bmatrix} & & 2 & 0 \\ & & 0 & 2 \\ 1 & 1 & & \\ 1 & 1 & & \end{bmatrix}$$

2nd Iteration :

$$D-D_1=\begin{bmatrix} (2) & 1 & 1 & 1 \\ 0 & 2 & (3) & 0 \\ 0 & 0 & 1 & (4) \\ 3 & (2) & 0 & 0 \end{bmatrix} \qquad D_2=\begin{bmatrix} 2 & 1 & & \\ & & 3 & 0 \\ & & 0 & 3 \\ 1 & 2 & & \end{bmatrix}$$

3rd Iteration :

$$D-D_1-D_2=\begin{bmatrix} 0 & 0 & (1) & 1 \\ 0 & (2) & 0 & 0 \\ 0 & 0 & 1 & (1) \\ (2) & 0 & 0 & 0 \end{bmatrix} \qquad D_3=\begin{bmatrix} & & 1 & 1 \\ 0 & 2 & & \\ & & 1 & 1 \\ 2 & 0 & & \end{bmatrix}$$

Schedule length =5+2+3+2=12=S

Example 2

$$T=D=\begin{bmatrix} (4) & 3 & 4 & 1 & 3 & 2 \\ 2 & 3 & 2 & 2 & 3 & (5) \\ 4 & (5) & 2 & 3 & 1 & 2 \\ 5 & 1 & (6) & 2 & 1 & 2 \\ 1 & 2 & 2 & (6) & 2 & 4 \\ 1 & 3 & 1 & 3 & (7) & 2 \end{bmatrix} \qquad S=17$$

STEP 3 n=6=3•2 k=3, m=2 The blocks are indicated on the matrix D

STEP 4 1rst Iteration

$$D_1=\begin{bmatrix} 4 & & & & & 2 \\ 2 & & & & & 4 \\ & 5 & 1 & & & \\ & 1 & 5 & & & \\ & & & 6 & 0 & \\ & & & 0 & 6 & \end{bmatrix}+\begin{bmatrix} & & 1 & 1 & & \\ & & 1 & 1 & & \\ & & & & 1 & 1 \\ & & & & 1 & 1 \\ 1 & & & & & \\ 1 & & & & & \end{bmatrix}+\begin{bmatrix} & & & & 1 & 2 \\ & & & & 2 & 1 \\ 2 & & & & & 1 \\ 1 & & & & & 2 \\ & & 1 & 2 & & \\ & & 2 & 1 & & \end{bmatrix}$$

2nd Iteration

$$D\text{-}D_1 = \begin{bmatrix} 0 & (2) & 3 & 0 & 1 & 0 \\ 0 & 2 & 1 & 0 & (2) & 1 \\ 2 & 0 & (1) & 2 & 0 & 1 \\ (4) & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 3 \\ 0 & 1 & 0 & 3 & 1 & 1 \end{bmatrix} \qquad D_2 = \begin{bmatrix} 2 & & 0 & & & \\ 0 & & 2 & & & \\ & 1 & & 1 & & \\ & 1 & & 1 & & \\ & & & & 0 & 2 \\ & & & & 2 & 0 \end{bmatrix}$$

3rd Itetation

$$D_3 = \begin{bmatrix} 0 & 2 & & & & \\ 2 & 0 & & & & \\ & & 2 & & & \\ & & 1 & & & \\ & & & & 1 & 1 \\ & & & & 1 & 1 \end{bmatrix} \qquad D_4+D_5 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Schedule length=6+2+3+2+2+2=17=S

Example 3

Here we consider the same traffic matrix as in example 2,but we write n as n=6=2•3 k=2 m=3

$$T=D= \begin{bmatrix} (4) & 3 & 4 & 1 & 3 & 2 \\ 2 & 3 & 2 & 2 & 3 & (5) \\ 4 & (5) & 2 & 3 & 1 & 2 \\ 5 & 1 & (6) & 2 & 1 & 2 \\ 1 & 2 & 2 & (6) & 2 & 4 \\ 1 & 3 & 1 & 3 & (7) & 2 \end{bmatrix} \qquad LS= \begin{bmatrix} a & b & c \\ b & c & a \\ c & a & b \end{bmatrix}$$

1rst Iteration

$$D_1=\left[\begin{array}{cc:ccc:c} 4 & 2 & & & & 2 \\ 2 & 2 & & & & 4 \\ 2 & 4 & & & & 2 \\ \hdashline & & 6 & 1 & 1 & \\ & & 1 & 6 & 1 & \\ & & 1 & 1 & 6 & \end{array}\right] + \left[\begin{array}{cc:ccc:c} & & 3 & 0 & 2 & \\ & & 0 & 2 & 3 & \\ & & 2 & 3 & 0 & \\ \hdashline 3 & 1 & & & & 1 \\ 1 & 1 & & & & 3 \\ 1 & 3 & & & & 1 \end{array}\right]$$

2nd Iteration

$$D_2=\left[\begin{array}{c:cc:cc} 1 & & & 0 & 0 \\ \hdashline 0 & 1 & 0 & & \\ 1 & 0 & 0 & & \\ \hdashline & 0 & & 0 & 1 \\ & 0 & & 1 & 0 \\ \hdashline 0 & 0 & 1 & & \end{array}\right] + \left[\begin{array}{c:cc:cc} 0 & 0 & 1 & & \\ \hdashline & 1 & & 0 & 0 \\ & 0 & & 1 & 0 \\ \hdashline 1 & 0 & 0 & & \\ 0 & 1 & 0 & & \\ \hdashline & 0 & & 0 & 1 \end{array}\right]$$

$$D_3=\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

S.L.=8+5+1+1=17=S

4.1 EVALUATION OF THE PROPOSED SA2

Schedule length

Theorem 8: The SA2 is an optimal algorithm.

proof: At each iteration the algorithm generates k sets of k nonconflicting blocks each. Each block in a set is a QDS matrix and all k nonconflicting blocks in the set,have equal line sums, say S_{ij} (ith set at the jth itetation). Thus the resulting traffic matrix after the jth iteration, $S-\sum_{j=1}^{J} D_j$ is also QDS with line sum $S-\sum_{j=1}^{J}\sum_{i=1}^{k} S_{ij} \geq 0$ if $J < W$, where W is the total number of itetations. This means that the schedule length cannot exceed S, the line sum of D. Because after W iterations $D-\sum_{j=1}^{W} D_j=0$ and and $S=\sum_{j=1}^{W}\sum_{i=1}^{k} S_{ij}$. Thus SA2 always achieves the minimum schedule length i.e. SA2 is optimal.

Number of switching

In a set of k nonconflicting blocks,where each block is a mxn latin square, the Number of switching is defined to be the maximum number of nonzero entries on any line of a block,in the set of k blocks. Thus the number of switching in a set of k blocks cannot exceed m. An example is given below:

$$A=\begin{bmatrix} 4 & 1 & 0 & 3 \\ 1 & 4 & 3 & 0 \\ 3 & 0 & 3 & 2 \\ 0 & 3 & 2 & 3 \end{bmatrix} \qquad B=\begin{bmatrix} 5 & 3 & 0 & 0 \\ 3 & 5 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

The traffic matrix A requires 3 switchings (2 for the first set and 1 for the second set of blocks), while B requires 2 switchings to transmit its traffic.

Theorem 9: The total number of switchings that require to transmit its traffic any nxn traffic matrix T, is bounded by n^2-2n+2.

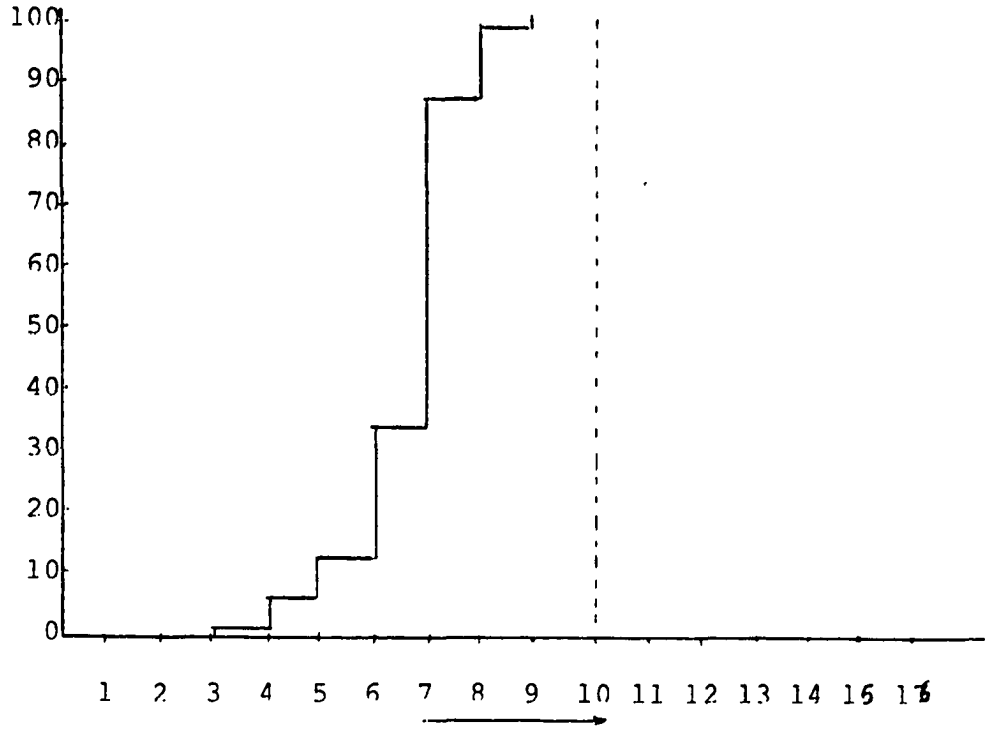
Proof: The number of switchings required to transmit its traffic,one set of k nonconflicting blocks with equal line sums is,at most,equal to m; and for all k sets is,at most, $n=k\cdot m$ switching. Now we may form a kxk latin square with each element correspoding to one block and each set to an SDR in it. Since at each iteration we don't repeat the same kxk latin square; using the same argument as in theorem 7 we imply that the number of switching, for a kxk traffic matrix would be bounded by k^2-2k+2. But since each switching of the kxk matrix is m switching for the nxn switching matrix matrix D_j the total number of number of switching is bounded by $(k\cdot m)^2-2(k\cdot m)+2=n^2-2n+2$.

In theorem 9 we have shown that the number of switchings of SA2 is bounded by the same bound as the number of mode matrices of SA1.

Simulation results

The SA2 has simulated for n=4, to verify the above theorem. Fig.2 shows the distribution curves for 100 randomly generated traffic matrices. The number of switchings,as shown in Fig.2,never exceeds the bound 10 for n=4,infact 86% of the traffic matrices have number of

switching less than 7.



Distribution curves for the number of required switching of the SA2 and for n=4

Fig. 4.1

Finaly the computational complexity of SA2 is reduced over SA1, since the max-min algorithm is applied fewer times to transmit the traffic matrix T. Also more traffic is transmited at the beginning of the frame. The above is based on the fact that while for SA1 we take the minimum over all n elements of the SDR, for SA2 this is done within the mxm block.

CHAPTER 2

1. MINIMIZING THE SCHEDULE LENGTH IN AN SS/TDMA SYSTEM WITH MINIMUM SWITCHINGS

In chapter 1 we studied the general problem of scheduling the traffic in an SS/TDMA system. There, the proposed algorithms, in order to achieve schedule length equal to its lower bound i.e., the critical line sum S, the number of mode martices or switchings z, had to be bounded by n^2-2n+2, for an nxn traffic matrix T.

In this chapter we intend to minimize the number of switchings or mode matrices, and also transmit continuously the message, consisting by the number of packets represented by the entries in T. The price we pay for that is slight longer schedule lengths, which as we shall see, is very well justified for the following reasons. a) The actual minimization of the schedule length is the minimization of the parameter L_{true} which is given by $L_{true}=L+z\cdot\tau$; where L is the schedule length without including the switch reconfiguration time τ. Basically τ may be the overhead due to switching, or to sychronization, guard time etc. Thus L_{true} depends on L and z, which also depend on each other, since when we try to minimize z, L cannot always be equal to the critical line sum S. Distiguishing two limiting cases we have: i) When τ is small in comparison with the packet

length the problem reduces to the one of chapter 1. ii) When τ is large in comparison with the packet length then L_{true} will be minimized when L is minimized subject to z being minimum. For this case we have developed two methods (sections 2,3) that satisfy both constrains and are also efficient. b) The continuous transmission of the message reduces the over all time delay of the system by reducing the mumber of the required acknowledgements each time. The above property may also be used in the case where the packet lengths are not fixed but variable, to save significant time of the schedule length.

The first method of combining objectives a) and b) is depicted in the following theorem.

Theorem 1 If in a given nxn traffic matrix T superimpose a nxn Latin Square (LS) and transmit T according to LS, then we always achieve a) Number of mode matrices less or equal to n. b) Continuous transmssion of every entry in T.

Proof: As shown in chapter 1 (theorem 4) any LS is a sum of n nonintersecting mode matrices and thus T may be decomposed according to the LS in at most n mode matrices (since one or more may include only zero entries in T). Also since the mode matrices of LS are nonintersecting every entry in T will belong to only one of them and so will be transmitted continuously. Example 1 illustrates the above theorem.

Example 1

$$T=\begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ b_2 & c_2 & d_2 & a_2 \\ d_3 & a_3 & b_3 & c_3 \\ c_4 & d_4 & a_4 & b_4 \end{bmatrix}=\begin{bmatrix} a_1 & & & \\ & & & a_2 \\ & a_3 & & \\ & & a_4 & \end{bmatrix}+\begin{bmatrix} & b_1 & & \\ b_2 & & & \\ & & b_3 & \\ & & & b_4 \end{bmatrix}+\begin{bmatrix} & & c_1 & \\ & c_2 & & \\ & & & c_3 \\ c_4 & & & \end{bmatrix}+\begin{bmatrix} & & & d_1 \\ & & d_2 & \\ d_3 & & & \\ & d_4 & & \end{bmatrix}$$

Considering now the generated schedule length by this method, it will be the sum of the maximum entries of each mode matrix. The schedule length of example 1 will be $L=a+b+c+d$ where $a=\max\{a_1,a_2,a_3,a_4\}$ $b=\max\{b_1,b_2,b_3,b_4\}$ $c=\max\{c_1,c_2,c_3,c_4\}$ $d=\max\{d_1,d_2,d_3,d_4\}$

Next we wish to minimize the schedule length L. This can be done by choosing the "appropriate" LS, among all the existing $n!(n-1)!(n-2)!\ldots$ latin squers, which will correspond to the minimum L. This problem is examined in section 2, where we propose an optimal algorithm that achieves the minimum possible schedule length. The proposed algorithm finds the appropriate LS, in polynomial time.

For an even further impovement on the minimum schedule length we use the method of nonconflicting blocks, which is the most appropriate with the objectives of this chapter. An example is given below comparing the schedule length of the above two methods.

Example 2

$$T=\begin{bmatrix} 9 & 4 & 2 & 5 \\ 2 & 2 & 3 & 4 \\ 5 & 4 & 3 & 2 \\ 6 & 5 & 3 & 5 \end{bmatrix}=\begin{bmatrix} 9 & & & \\ & & 3 & \\ & 4 & & \\ & & & 5 \end{bmatrix}+\begin{bmatrix} & & 2 & \\ & & & 4 \\ 5 & & & \\ & 5 & & \end{bmatrix}+\begin{bmatrix} & 2 & & \\ & & & 5 \\ & 3 & & \\ 6 & & & \end{bmatrix}+\begin{bmatrix} & 4 & & \\ 2 & & & \\ & & & 2 \\ & & 3 & \end{bmatrix}$$

↑ crit line sum=22

Min possible schedule length=9+5+6+4=24

$$T = \begin{bmatrix} 9 & 4 & 2 & 5 \\ 2 & 2 & 3 & 4 \\ 5 & 4 & 3 & 4 \\ 6 & 5 & 3 & 5 \end{bmatrix} = \begin{bmatrix} 9 & & 2 & \\ 2 & & 3 & \\ & 4 & & 2 \\ & 5 & & 5 \end{bmatrix} + \begin{bmatrix} & 4 & & 5 \\ & 2 & & 4 \\ 5 & & 3 & \\ 6 & & 3 & \end{bmatrix} = \begin{bmatrix} 11 & & \\ & 10 & \end{bmatrix} + \begin{bmatrix} & & 9 \\ 11 & & \end{bmatrix}$$

Block method: Schedule length = 11+11 = 22 = S

Using the same traffic matrix in the second case as in the first one, and choosing the appropriate set of nonconflicting blocks, we succeeded to reduce the schedule length to its lower bound (critical line sum) although is not always possible.

The method of nonconflicting blocks is implemented by an heuristic algorithm given in section 4, which generates the shortest the shortest schedule length for the 4x4 case, as shown by simulation. However this method reduces to the firt one, when n is not a prime number, thus it may be considered as an extension of the fist method (theorem 1).

Related work can be found in [21] and [22]. In [22] Natarajan and Calo address the problem of minimum switchings in a similar manner, there they examine the lower and upper bounds on the schedule length. They show that the ratio of the minimum schedule length with the constrain that only n switchings are used to the unconstrain minimum is unbounded. Also they propose three heuristic algorihtms. In [22] Gopal and Wong examine the problem of minimum switchings in a slightly different manner and they show its NP-completness (see[28]), they also propose a heuristic algorithm.

2. THE MINIMUM SCHEDULE LENGTH (MSL) ALGORITHM

The algorithm we propose here is based on a backtracking method of constracting an "appropriate" Latin Square (LS) which superimposing to the traffic matrix T generates the minimum possible schedule length. Thus MSL algorithm is optimum is this sense.

Let L be a LS needed to achieve the min. schedule length. Then according to the algorithm all the 1s will be chosen first then all the 2s, in the remaining n^2-n positions etc. If no suitable position can be found for the mth element ($1 \leq m \leq n$), then we "backtrack" by finding an alternative choice for any of the prior elements which permits us to fill up the mth element of the LS. But for precise description of the algorithm we need the following definition.

$$L = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 \end{bmatrix}$$

Degree Of Freedom (DOF) of a line of n elements, given k of them chosen ($k<m$) and also given a threshold t_m, is equal to the number of elements that are less or equal in magnitude to t_m and are among the (n-k) that haven't been chosen. For example if a line is [(1),(2),3,4,2] with 1,2 chosen and a threshod 3 the degree of freedom is 2.

The Algorithm

STEP 0: Given a traffic matrix T, choose a critical line of T and set its elements in increasing order i.e., assign to each entry of the line a number m ($1 \leq m \leq n$) where m=1 takes

the smallest entry, (if two entries are equal assign the smallest number to the one with the smaller DOF of the corresponding cross line). Let t_m be the entry to which the number m is assigned, then write the element m in the corresponding possition of LS.

STEP 1: Initialize by setting m=1, r=1

a) Delete the row and column of the element t_m in T. Let T_m be the resulting (n-1)x(n-1) matrix.

b) Write the initial DOF of the rows in T_m if the critical line is a column (or DOF of the columns if the critical line is a row), given the threshold t_m.

c) Choose the row (column) with Minimum Degree of Freedom (MDF). If MDF is not zero go to step 3a).

d) If MDF=0 Then we have either to backtrack or increase threshold t_m. When backtracking is possible (see step 2a) and increase of threhold is not necessery (see below), then we always backtrack by cancelling the element (m-1), which always permits us to have MDF>0 for the current element m. (The above backtracking process might go back several elements. This chain some times may be avoided by going back directly to an element k (k<m), an alternetive position of which will give us MDF>0. This element can be determined by keeping track of the MDFs of all k<m, and taking the one with the maxMDF.) When backtracking is not possible we have to increase the threshold t_m of the mth element. We always have to increase the threshold when MDF becomes zero at m=1 or at any m for which the row with MDF=0 has all unoccupied

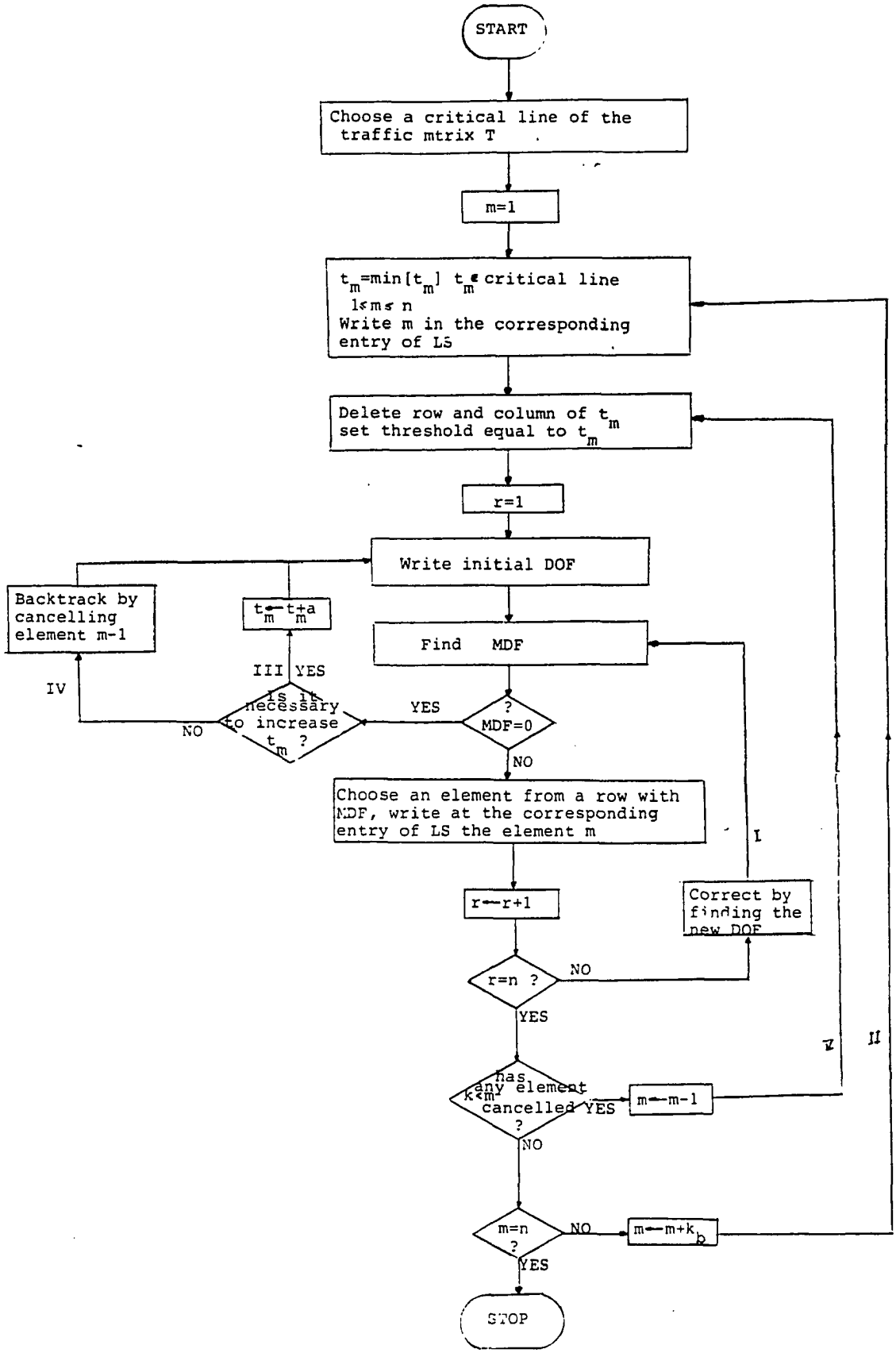
elements greater than t_m. Backtracking may also lead to a threshold increase which is smaller or equal to the one required at element m. For this reason, we also keep track, at this step, of the required threshold increase of each element k. Set new threshold $t_m \leftarrow t_m + a$ $a \geq 0$. Go to step 1b).

STEP 2 a) Choose any random entry from the row (column) with MDF which is unoccupied and which is smaller than the threshold. Write in the corresponding position of LS the element m. At this step we also keep track of whether this element can be backtracked. If this element has MDF>1 then backtracking is possible i.e., we may cancel it, from this row on, and also all the following elements, if necessary, to find an alternative position in LS. The number of alternative positions from this row on, is at most equal to MDF.

b) If the element m hasn't been written on every row and (column) i.e., $r<n$ set $r \leftarrow r+1$. Find the new DOF of the remaining rows (columns) by deleting the rows and columns with the element m. Go to step 1c).

d) If element m has been written on every row of LS ($r=n$), check whether any of the previous elements k, ($k<m$) has been cancelled. If it has set $m \leftarrow k$ and go to step 1b). If it hasn't set $m \leftarrow m+k_b$ (which is the next element to be filled up in LS) and go to step 1b). If $m=n$ stop, LS is complete.

Next a flow chart of the MSL algorihtm is shown, also illustrative example is given.



The MSL Algorithm Flow Chart

Fig 1

Example

$$T=\begin{bmatrix} \{3\} & 2 & 3 & (1) & 4 \\ [3] & 5 & 1 & 2 & (1) \\ \langle 3\rangle & 2 & (3) & 1 & 3 \\ (3) & 3 & 1 & 5 & 2 \\ 4 & (1) & 2 & 2 & 2 \end{bmatrix} \begin{matrix} -3 \\ -3-2 \\ -4-3-2 \\ \\ -4-3-2-1 \end{matrix} \quad m=1 \nearrow$$

$l = 1,2,3,4$

crit. line sum=16

$m=1 \rightarrow ()$ $m=2 \rightarrow []$ $m=3 \rightarrow \{\}$ $m=4 \rightarrow \langle\rangle$

The degrees of freedom of each iteration for m=1 are shown to the right of T (l=r-l=1,2,3,4). Each time we choose an element from the row with MDF, thus we have chosen the first element of LS. With the same manner we choose elements m=2,3.

At the 4th element MDF becomes zero, and since backtracking is possible we erase element k=1, which is chosen to avoid altering all the elements k<m, and we try again m=3.
After selecting m=4 we go back to m=1 at the selection of which we dont need to backtrack again.

$$T=\begin{bmatrix} \{3\} & [2] & 3 & (1) & 4 \\ [3] & 5 & \langle 1\rangle & \{2\} & (1) \\ \langle 3\rangle & \{2\} & (3) & 1 & [3] \\ (3) & \langle 3\rangle & [1] & 5 & \{2\} \\ 4 & (1) & \{2\} & [2] & \langle 2\rangle \end{bmatrix} \begin{matrix} -1 \\ -1-0 \\ \\ -1-1 \\ -1-1 \end{matrix} \quad m=4 \nearrow$$

$$T=\begin{bmatrix} \{3\} & [2] & 3 & \langle 1\rangle & 4 \\ [3] & 5 & \langle 1\rangle & \{2\} & 1 \\ \langle 3\rangle & \{2\} & 3 & 1 & [3] \\ (3) & \langle 3\rangle & [1] & 5 & \{2\} \\ 4 & 1 & \{2\} & [2] & \langle 2\rangle \end{bmatrix} \begin{matrix} -2-2-2-1 \\ -2-2-1 \\ \\ -1 \\ -2-1 \end{matrix} \quad m=4 \nearrow$$

$$T=\begin{bmatrix} 3 & [2] & (3) & \langle 1\rangle & 4 \\ [3] & 5 & \langle 1\rangle & \{2\} & (1) \\ 3 & \{2\} & 3 & (1) & [3] \\ (3) & \langle 3\rangle & [1] & 5 & \{2\} \\ 4 & (1) & \{2\} & [2] & \langle 2\rangle \end{bmatrix} \begin{matrix} -1-1-1 \\ -1-1 \\ -2-2-2-1 \\ \\ -1 \end{matrix} \quad m=1 \nearrow$$

At the selection of the 5th element we had to increase the threshold because MDF became zero and the uncuppied element at the line with MDF was greater than the threshohold.

$$LS=\begin{bmatrix} 3 & 2 & 1 & 4 & 5 \\ 2 & 5 & 4 & 3 & 1 \\ 4 & 3 & 5 & 1 & 2 \\ 1 & 4 & 2 & 5 & 3 \\ 5 & 1 & 3 & 2 & 4 \end{bmatrix}$$

The resulting schedule length is $(3)+[3]+\{3\}+\langle 3\rangle+5=17$

Theorem 2 MSL algorithm is an optimal algorithm i.e., always achieve the minimum possible schedule length under this formulation of the problem (theorem 1).

Proof Let S be the critical line sum of the nxn traffic matrix T. Then S is the lower bound of the schedule length L ($S \leq L$). Schedule length L=S will be achieved if no increase of the initial threshold occurs for all m, because S is the sum of the elements of the critical line. If an increase of the threshold occurs, by h_m for the mth element ($1 \leq m < n$), the schedule length will be $L = S + \sum_{m=1}^{n} h_m$ ($h_m \geq 0$). We will show that this increase $\sum_{m=1}^{n} h_m$ is always the minimum possible. According to the algorithm when MDF becomes zero, we always increase the threshold only when is necessery and by the minimum amount to make MDF=1, and this happens in the following cases: a)When m=1 because no backtracking is possible. The increase will be the minimum possible to make MDF=1. b)When all unoccupied elements of the row (column) with MDF=0 are greater than t_m. In this case t_m will be increased to the minimum of them, thus h_m will be minimum and so does $\sum_{1=1}^{m-1} h_1$ since $t_{m-1} \leq t_m$. c)If the purpose of backtracking is to reduce the increase of threshold required at element m, this increase will be minimum since we keep track of the increaces required for all k<m. Thus $\sum_{m=1}^{n} h_m$ will always be minimum.

The computational Complexity of MSL Algorithm

As we shall show MSL algorithm is a polynomial algorithm. This mainly comes from the fact that each time we backtrack, according to the algorithm, we only alter once the position of any elemnt $k < m$ and we never try exhaustive search of all the alternatives which would lead to an exponentially type algorithm.

More specifically for an nxn traffic matrix if no backtracking is possible (or not needed) the maximum number of iterations of loops I, II and III (see fig 1) will be n^2 if no increase of threshold is required (loops I, II). If an increase of threshold is required n times, for each element m ($1 \leq m \leq n$), then the total number of iterations (loop I) will be

$$n \cdot \sum_{r=1}^{n} r = n\lceil n(n+1)/2 \rceil = \lceil n^3+n^2 \rceil/2 \quad (1)$$

because we assumed MDF becomes zero at every r, which is a very extreme case and very rare. Now if one backtracking is required (loops I, IV, and V), say from the mth element back to element 1, then the total number of iterations will be $n^2+m(2n)$, where on the backtracking term m(2n), the number 2 comes from the assumption that MDF becomes zero at r=n-1. Now although, backtracking is needed usually only once or twice during the whole process of constructing LS, we consider the extreme case where we backtrack at every element m ($1 \leq m < n$), then the total number of interations will be

$$2n \sum_{m=1}^{n} m = 2n\lceil n(n+1)/2 \rceil = n^3 + n^2 \qquad (2)$$

Next, since we either backtrack or increase threshold (not both) when MDF=0 we take the maximum between (1) and (2). Thus the upperbound on the number of iterations is $(n^3 + n^2)$, where each iteration represents basically loop I i.e., find and choose the row with MDF and correct by finding new DOF of each of the remaining rows.

Simulation results regarding the generated schedule length of MSL algorithm in comparison with the critical line sum will be presented in section 3.

3. A SUBOPTIMUM ALGORITHM

A simplified version of MSL algorithm which also reduces its complexity is the following suboptimum one.

Given a traffic matrix T we construct a matrix T_s which consists of the critical line of T and the line with line sum next smaller to the critical line (subcritical line) and replacing all the other elements of T with zeros. Also, if there is any element in T, greater than the maximum element of the critical line we include it in T_s. Next we apply MSL algorithm on T_s to find a latin square according to which the matrix T_s will be transmitted.

Example

$$T = \begin{bmatrix} 5 & 7 & 0 & 3 \\ 4 & 2 & 2 & 3 \\ 3 & 3 & 3 & 6 \\ 5 & 3 & 3 & 4 \end{bmatrix} \longrightarrow T_s = \begin{bmatrix} 5 & 7 & 0 & 3 \\ 4 & 0 & 0 & 3 \\ 3 & 0 & 0 & 6 \\ 5 & 0 & 0 & 4 \end{bmatrix} \longrightarrow LS = \begin{bmatrix} 3 & 4 & 2 & 1 \\ 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{bmatrix}$$

17 16

4. THE METHOD OF NONCONFLICTING BLOCKS

Here we propose a heuristic algorithm based on the method of transmitting nonconflicting blocks indepedently and simultaneously, which is also used in SA2 (chapter 1).

The algorithm although heuristic, it suceeds to reduce the schedule length compared with the previous methods used in this section. Unfortunally this method only can be applied when n is not a prime number and in combinations with the previous algorithms.

The Algorithm has as follows: Given an n x n Traffic matix T

STEP 1: Write n as a product $n=k\cdot m$ where k=The number of nonconflicting blocks in a set, and m=size of each block (k,m integers)

STEP 2: Find the first set a k nonconflicting block as follows

a) Divide the critical line of T, into k disjoint sets, with m elements each set, so that, the difference between the sum of the elements of each set is minimum.

b) Pick up one of the above sets and form an mxm block so that its critical line sum does not exceed the sum of the elements in the set. If this is not possible, we choose the elements of the block so that the difference between its critical line sum and the sum of the elements in the set is minimized.

c) We delete the rows and columns of the first block and we

apply steps 2a, and 2b on the resulting matrix to find the remaining blocks in the set. (Between the critical line sums of each block in a set of nonconflicting blocks, the maximum one must be the one that belongs to the first block i.e., the sum of the elements given in the set. If the above is not possible, we try to minimize the difference).

STEP 3: a) Given the first set of k nonconflicting blocks, we determine the remaining k^2-k blocks (or cells) in T by taking the intersection of the rows and columns of every two blocks of the first set.

b) Form a kxk matrix L each entry of which represents the critical line sum of the corresponding block in T. The main diagonal in L may be used to represent the first k blocks.

STEP 4: Apply the MSL algorithm (section 1) in L to find a latin square which minimize the schedule length, given the element 1 in LS is filled up. So this way we define the remaining (k-1) sets of nonconflicting blocks in T with the aid of the latin square found above.

STEP 5: Transmit each block in a set of nonconflicting blocks simultaneously and independently with each other, and one set after the other. In order to transmit an mxm block optimally $m \geq 3$ we need to apply the algorithm of section 1 or 2

Example is given below

Example

$$T=\begin{bmatrix} 1 & (4) & (2) & 5 & 6 & 9 \\ 8 & 4 & 11 & 7 & (2) & (2) \\ [3] & (6) & (6) & [12] & \{9\} & \{5\} \\ (5) & 5 & 7 & (7) & 7 & 9 \\ 4 & 11 & 4 & 3 & (8) & (1) \\ (9) & 4 & 3 & (6) & 7 & 9 \end{bmatrix} \leftarrow \text{crit line sum=41}$$

n=6, k=3(number of sets),m=2(block size)

Set 1→(), 2→[], 3→{}

Step 2c: By cancelling the rows the rows and columns of the first bock of the 1st set and appling again steps 2a and 2b on the remaining matrix we determine the remaining blocks of the first set.See matrix M below.

$$T\longrightarrow M=\begin{bmatrix} 8 & 7 & (2) & (2) \\ (5) & (7) & 7 & 9 \\ 4 & 3 & (8) & (1) \\ (9) & (6) & 7 & 9 \end{bmatrix} \leftarrow 31$$

By taking the inrsection of the rows and columns of every two blocks of the first set we determine the blocks of the remaining sets(step 3a). Representing each by its crit. line sum we form the kxk matrix L. The main diagonal in L represents the first set. (step 3b).

crit line sum=55

$$L=\begin{bmatrix} (12) & 17 & 15 \\ 12 & (15) & 15 \\ 15 & 18 & (10) \end{bmatrix} \longrightarrow LS=\begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{bmatrix}$$

The resulting schedule length is 55

5. SIMULATION RESULTS

In order to compare the average schedule length generated by each of the presented algorithms we have simulated each one of them and run it on 100 randomly generated traffic matrices. Each entry of the traffic matrix, that represents the message length in packets, have the same probability distribution which is chosen to be in case i) uniform distributed between 0 and 20, and in case ii) Poisson ($P(x=k)=e^{-\lambda}(\lambda^k/k!)$) with mean $\lambda=7$. MSL algorithm and Suboptimum algorithm have simulated for n=4,5 and 6, while the The Method Nonconflicting Blocks has simulated for n=4. The results are shown in Tables 1 and 2, where $\bar{S}$=Average critical line sum, $\bar{L}$=average schedule length and $\bar{L}_o\% = (L-S)100/S$ is the average percentage overlength.

Traff. matrix	$\bar{S}$	$\bar{L}$	$\bar{L}_o\%$
MSL algorithm			
4x4	54.130	58.070	7.2%
5x5	68.450	74.850	9.3%
6x6	83.747	91.613	9.4%
Suboptimum algorithm			
4x4	53.78	60.770	13.0%
5x5	68.50	81.950	19.6%
6x6	83.75	100.4	19.9%
Method of nonconflicting blocks			
4x4	54.25	57.59	6.1%

Entries Uniformly dist. between 0-20

Table 1

Traff. Matrix	$\bar{S}$	$\bar{L}$	$\bar{L}_o$%
MSL algorithm			
4x4	36.23	38.02	4.8%
5x5	44.99	48.55	7.9%
6x6	54.81	59.13	7.9%
Suboptimum algorithm			
4x4	36.07	39.40	9.2%
5x5	44.87	50.94	13.5%
6x6	54.77	62.49	14.1%
Method of nonconflicting blocks			
4x4	36.23	38.05	4.8%

The traff. matrix has Poisson entries with mean 7

Table 2

As we observe from the tables 1 and 2, the Method of nonconflicting blocks gives us the shortest averare overlength for the 4x4 case. Next comes the MSL algorithm which gives the minimum possible L under the corresponding scheduling method (theorem 2). The average percentage overlength of this method in both cases (uniform and poisson) is within 5 to 10% of the $\bar{S}$, and this the price we pay for the advantages we get. However 5 to 10% is not much if we consider that it may cost us much more in schedule length with number of switchings bounded by n^2+2n+2. Finally the suboptimum algorithm give us an $\bar{L}_o$% within 10 to 20% which is comparable with the heuristic algorithm given by Gopal and Wong in [21].

CHAPTER 3

1. A Class of Tree Algorithms With Variable Message Length

A very efficient type of protocols for satellite channels are the tree protocols. They have the advantage that they are stable and carry small delays. The maximum throughput is 0.347 packet/slot for the basic and 0.43 for the dynamic algorithms, references [31] and [32]. In the tree algorithms the channel time is slotted and all users are spatially isolated, i.e. the only means of communication is through the channel itself. The collision, among a number of users, is resolved precisely after a certain period of time which is called Collision Resolution Interval (CRI). All users in the system, whether involved in the collision or not, are aware of the time span of the CRI.

To improve the throughput, we allow the user to transmit the remaining of its message after successful transmission of the first packet of the message. This is analyzed in sections 2 and 3, for small and large numbers of users, respectively.

The small number of users analysis is based on a Markov chain which has as state probabilities the number of users collided at the beginning of each CRI. The large number of users analysis is based on the traffic independent properties of the tree algorithms given in [32]. In both cases we get considerable improvement on the maximum throughput.

In section 4 the idea of "reservation" is utilized as follows: The collision among the users is resolved utilizing small "test" pockets which carry the information about the length of each user's message

contending for transmission. After the end of the test packet's CRI the channel time is reserved for each user, one after the other, to transmit its message. This scheme, named Reservation Tree Algorithm, has maximum throughput approaching one as the message length increases, and also carries small delays. Thus utilizing the idea of reserving the channel by means of tree algorithm and transmitting the message afterwards, we have the good properties of the tree algorithm and we also achieve high maximum throughput. A comparable scheme that utilizes the idea of transmitting the whole message after successfully contending for its first packet is the Reservation - ALOHA protocol presented in [35].

In section 5 of this chapter, the problem of long propagation delays of satellite channels is considered. The proposed interleaving scheme, named Alternating Tree Algorithms, has certain advantages over the existing schemes [34], adapts efficient satellites channel long propagation delays.

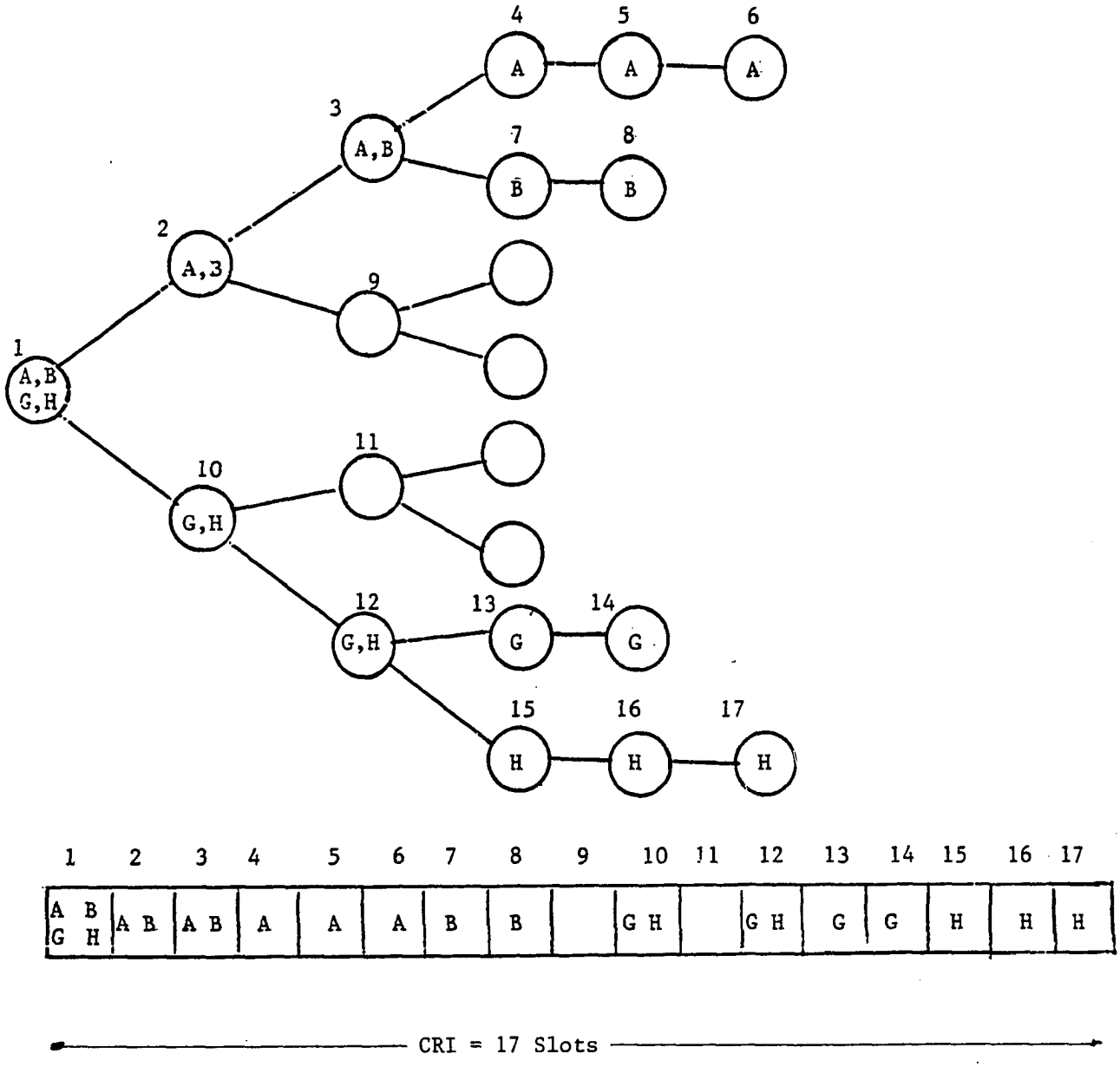
Finally, in section 6 we present and analyze a hybrid TDMA-tree scheme appropriate for multi-beam satellites according to it, each TDMA slot is dedicated for collision resolution (according to the tree algorithm) among the users with the same destination zone.

2. Case of Small Number of Users

We assume a noiseless channel of the time slotted type. The length of each slot equals to the length of the packet T. Each user generates a message of h packets in a slot, with probability σ. The length of the message is assumed to be geometric with parameter p. The user can have a maximum of one message, including the collided one. The feedback information is immediately available at the end of each slot (zero propagation delay). This information is: a) empty (no packet transmitted); b) successful (one packet transmitted); c) collided (two or more packets transmitted).

The basic Tree-algorithm can be described as follows: each user is assigned a leaf in a binary tree. After a collision, those users who own a leaf in the upper half of the tree retransmit in the next slot and those who own a leaf in the lower half retransmit in the next slot after the collision (if any) among those in the upper half has been resolved.

For the case of a message of h packets long, the first packet in the message is the one that goes through the collision. Once it succeeds in transmission, it is followed by the rest of the message (h-1 packets). The last packet in the message has a flag which indicates the end of the message. An illustrative example is given below; we consider a system of 8 users (M=8), Fig. 2.1. Users A, B, G, and H have a message to transmit. Users A and H have 3-packet messages, while users B and G have 2-packet messages. After the first packet of user A succeeds in slot 4, user A continues transmitting the rest of the message in slots 5 and 6. The third packet contains a flag indicating the end of the message. User B then transmits his 2-packet message in slots 7 and 8. The collision resolution interval (CRI) in this example is 17 slots.



The Tree Algorithm with Variable Message Length (8 users)

Fig. 2.1

The analysis here will be based on obtaining P_j, the steady state probability of having j users contending for transmission at the beginning of the CRI.

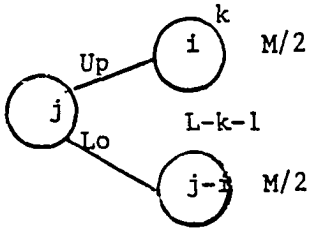
Since P_j depends on the CRI length we need to obtain the conditional distribution of the CRI.

2.1 Conditional Distribution of the Collision Resolution Interval (CRI) Length

Let $\alpha(L/h_1,h_2,\ldots,h_j,M)$ = steady state probability that the CRI length is L slots long given j users contending for transmission, each with message length $h_1,h_2,\ldots,h_j$ respectively, and M is the total number of users in the system.

In this section $\alpha(L/h_1h_2\ldots,h_j,M)$ is obtained as a solution of a recursive equation.

The probability that exactly i of the colliding j users own a leaf on the upper part of the tree and exactly (j-i) own a leaf in the lower part (see Fig 2.2) is the hyper geometric probability $r_i(j,M)$



$$r_i(j,M) = \frac{\binom{M/2}{i}\binom{M/2}{i-j}}{\binom{M}{j}} \qquad (2\text{-}1)$$

$$o \leq i \leq M/2 \quad \text{for} \quad j < M/2$$

$$j-M/2 \leq i \leq M/2 \quad \text{for} \quad j < M/2$$

Fig. 2.2

If k is a random variable representing the number of slots needed to resolve the collision among the i users following the branch Up, then L-k-1 is the number of slots needed to resolve the collision among the (j-1) users following the branch Lo

$$\alpha(L = 0/J = 0,M) = 1 \text{ and } \alpha(L = h/h,M) = 1$$

and for $M \geq 8$ and $4 \leq j \leq M/2$ the following recursive equation can be written

$$\alpha(L/h_1,h_2,\ldots,h_j,M) = \{r_o(j,M) + r_j(j,M)\}\alpha(L-2/h_1,\ldots,h_j,M/2) +$$

$$+ r_1(j,M)\alpha(L-h_1-1/h_2,\ldots,h_j,M/2) + r_{j-1}(j,M)\alpha(L-h_j-1/h_1,h_2,\ldots,h_{j-1},M/2) +$$

$$+ \sum_{i=1}^{j-2} r_i(j,M)\{\sum_{k=k1}^{k_2} \alpha(L=k/h_1,\ldots,h_i,M/2)\alpha(L=k-1/h_{i+1},\ldots,h_j,M/2)\} \qquad (2.2)$$

for $L = j+\sum_{i=1}^{j} h_i-1,\ldots,\hat{L}(M/2,h_1,\ldots,h_j)$

The above recursive equation represents the summation, over i, of the possibility having i users following branch Up times the summation, over k, of the probability that the time to resolve the contention of the i users, in a tree of M /2 leaves, is k slots, times the probability of resolving the j-i users, in a tree of M /2 leaves, in L-(k+1) slots.

k_1 is the minimun and k_2 the maximum value of k given i users contending out of M/2

$$k_1 = 2i-1+\sum_{n=1}^{i} (h_n-1) = i + \sum_{n=1}^{i} h_n-1$$

$$k_2 = L - \{2(j-i) + \sum_{n=i+1}^{j} (h_n-1)\} = L - (j-1) - \sum_{n=i+1}^{j} h_n$$

$\hat{L}(j,h_1,h_2,\ldots,h_j)$ is the maximum length of the CRI with j users contending and each with message length $h_1,h_2,\ldots,h_j$ respectively.

For $\dot{J} = M/2+1$ $\quad r_o(j,M)=r_j(j,M)=0$ $\quad$ and $i=2,\ldots,M/2-1$

$j = M/2+2$ $\quad r_o(j,M)=r_1(j,M)=r_{j-1}(j,M)=r_j(j,M)=0$

and $i=j-M/2\ldots,M/2$

The following special cases are examined below.

For $J=3$ $\quad M\geq 8$

$$\alpha(L/h_1,h_2,h_3,M)=\{r_o(3,M)+r_3(3,M)\}\alpha(L-2/h_1,h_2,h_3,M/2) +$$
$$+ r_1(3,M)\alpha(L-h,-1/h_2,h_3,M/2) + r_2(3,M)\alpha(L-h_3-1/h_1,h_2,M/2)$$

For $M=4$

$j=4$ $\quad \alpha(3+\sum_{i=1}^{4} h_i/h_1,\ldots,h_4,M=4)=1$

$J=3$ $\quad \alpha(2+\sum_{i=1}^{3} h_i/h_1,h_2,h_3,M=4)=1$

$j=2$ $\quad \alpha(h_1+h_2+1/h_1,h_2,M=4)=r_1(2,4)=2/3$

$\quad \alpha(h_1+h_2+2/h_1,h_2,M=4)=ro(2,4)+r_2(2,4)=1/3$

$j=1$ $\quad \alpha(h/h,M)=1$ $\qquad$ $j=0$ $\quad \alpha(L=1/0,M)=1$

2.2 The System Markov Chain

If we consider as the state of the system the number of users contending for transmission at the beginning of the CRI, then we can form an imbedded Markov chain describing the system. Let's define P_j = steady state probability of having j users contending for transmission at the beginning of CRI.

If α_L is the steady state probability that CRI length is L, then

$$\alpha_L = \sum_{h_j} \cdots \sum_{h_2} \sum_{h_1} \alpha(L/h_1,h_2 \cdots h_j,M)P(h_2)\cdots P(h_j) \qquad 2.3$$

$$L = 3,4,\ldots,jM + H-1 \text{ and } L \geq j + H-1 \text{ for all } j$$

$$\text{and } j = M \text{ for } L \geq 2M-1$$

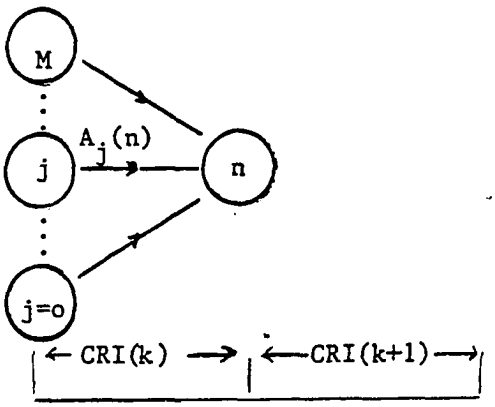
where in the above equation $P(h_i)$ $i = 1,\ldots,j$ are the probabilities that the message lengths are h_i packets long and

$$H = \sum_{i=1}^{j} h_i$$

We assume $P(h_i=k) = pq^{k-1}$ and $q = 1-p$ (geometric distribution)

Let $A_j(n)$ represent the transition probability for the system Markov Chain, from state j at time (t-L) to state n at time t (see fig. 2.3).

Fig. 2.3



The steady state probability P_n of having n users contending for transmission at the CRI(k+1) is given by

$$P_n = \sum_{j=o}^{M} A_j(n)P_j \qquad o \leq n \leq M \qquad (2.4)$$

Let σ_L = prob (a user generates at least one message during a CRI of length L)

$$\sigma_L = 1 - (1-\sigma)^L$$

where σ is the message arrival rate (i.e., σ messages/slot).

The transition probabilities then are given by

$$A_j(n) = \sum_{h_j} \cdots \sum_{h_2} \sum_{h_1} \sum_{L=j+H-1}^{\hat{L}(j,M)} \binom{M}{n} \sigma_L^n (1-\sigma_L)^{M-n} \alpha(L/h_1,h_2,\ldots h_j,M) P(h_1)P(h_2)\ldots P(h_j)$$

where $L(j,M) = \hat{L}_{max}(j) + H-j$ and $\hat{L}_{max}(j)$ is the maximum CRI length if each of the j collided users had a single packet message . In other words, $A_j(n)$ is the probability that each of the n users, out of M, generate at least one message during a CRI of length L. If more than one message arrives during a CRI, one will be stored in the user's buffer to be transmitted in the next CRI while the rest will be rejected.

2.3 Delay Analysis

The average total message delay is the sum of the average waiting time until transmission at the beginning of the next CRI, denoted by W, plus the average time spent from the beginning of the CRI until successful transmission of the message. We refer to the latter as service time S.

$$D = W + S \tag{2.6}$$

W is equal to one half of the average CRI length $\bar{L}$, i.e.

$$W = 1/2\ \bar{L} \quad \text{and} \quad \bar{L} = \sum_{L=1}^{\infty} L\ \alpha_L \tag{2.7}$$

since we assume on the average the packet arrives at the middle of the CRI. The service time is derived similarly as in reference [33]

Let $W(K/h_1,h_2,\ldots,h_j,M)$ = Prob (a packet takes K slots from the moment of its first transmission until successful transmission of the message, given j colliding users each with message length h_i packets $1 \leq i \leq j$ and total number of M users)

$$W(k/h,M) = \begin{cases} 1 & k=1 \\ 0 & k\neq 1 \end{cases} \qquad W(0/0,M) = 0$$

Now we obtain $W(k/h_1,h_2,\ldots,h_j,M)$ in a recursive approach as follows:

for $M\geq 4$ and $j\leq M/2$

$$W(k/h_1,h_1,\ldots,h_j,M) = r_o(j,M)W(k-1/h_1,\ldots,h_j,M/2) + \tag{2.8}$$

$$+ \sum_{i=1}^{j} r_i(j,M)\{(\tfrac{i}{j})W(k-1/h_1,..,h_i,M/2)+(\tfrac{j-1}{j})\sum_{L}\alpha(L/h_1,\ldots,h_i,M/2)W(k-L-1/h_{i+1},..,h_j,M$$

where $r_i(j,M)$ is defined in equation (2.1) and $\alpha(L/h_1,\ldots,h_i,M/2)$ is given from equation (2.2).

The above equation represents a summation over i, of the probability of having i users following branch Up (see Fig. 2.3). $r_i(j,M)$, times the probability the packet takes (k-1) slots given i colliding users in the Tree of M/2 leaves. The second term inside the summation represents the case when the packet is one of the (j-i) packets that followed branch Lo. In this case the message has to wait until all the i messages have been transmitted. $\frac{j-i}{j}$ is the probability that the packet is one of those which followed branch Lo, times the summation over L of the probability the CRI length is L slots long given i colliding users in the upper subtree, times the probability a packet takes k-L-1 slots given L-1 colliding users in the lower subtree of M /2 leaves.

For $j>M/2$

$$W(k/L_1,..,h_i,M) = \sum_{i=j-M/2}^{M/2} r_i(j,M)\{(\tfrac{i}{j})W(k-1/h_1,..,h_i,M/2) +$$

$$+ (\tfrac{j-i}{j})\sum_{L}\alpha(L/h_1,\ldots,h_i,M/2)W(k-L-1/h_{i+1},\ldots,h_j,M/2)\} \tag{2.9}$$

By unconditioning over j and h_i we get

$$W(k) = \sum_{h_j}\cdots\sum_{h_2}\sum_{h_1} W(k/h_1,h_2,\ldots,h_j,M)P(h_1)P(h_2)\ldots P(h_j)/1-P_o \tag{2.10}$$

We divide by $(1-P_0)$ since we assume at least one packet is transmitted at the beginning of the CRI.

Hence the average service time S is given by

$$S = \sum_{k=o}^{\hat{L}(M)} K\, W(k) \qquad (2.11)$$

Now if we consider that every user has the same probability distribution for its message length, i.e., $P(h_1)=P(h_2)=\ldots=P(h_j)=P(h)$, that is geometric $p(h=k)=q^{k-1}p, k=1,2,..,p=1-q$, and $\bar{h}=1/p$, then we may represent all the users having the same message length random variable h, which greatly simplifies the computational complexity of the problem.

Thus instead of having $\alpha(L/h_1,..,h_j,M)$ we have $\alpha(L/j,M,h)$ the steady state probability that the CRI length is L slots long, given j users contending for transmission, M users in the system and the message length is h packets. Also $W(k/h_1,..,h_j,M)$ is replaced with $W(k/j,M,h)$ the probability a packet takes K slots from the moment of its first transmission until successful transmission given j, M, and h.

The Delay vs. Throughput characteristics are given in Fig. 2-4 for M=4 users and average message length $\bar{L}$ = 1, 2, and 6. The Throughput Thr has been defined as follows:

$$\text{Thr} = \frac{\text{number of successful transmissions per CRI}}{\text{total number of slots per CRI}}$$

Thus the throughput can be expressed as follows:

$$\text{Thr} = \frac{\bar{h} \cdot \bar{j}}{\bar{L}} \qquad (2.12)$$

where $\bar{j} = \sum_{j=0}^{M} jP_j$ is the average number of users collided in the first slot of the CRI and $\bar{L}$ the average CRI length. With M=4 users and average message length $\bar{h}$ = 1,2, and 6. The maximum throughput is 0.58, 0.73, and 0.89, respectively.

Delay vs Throughput Characteristics

M = 4 Users

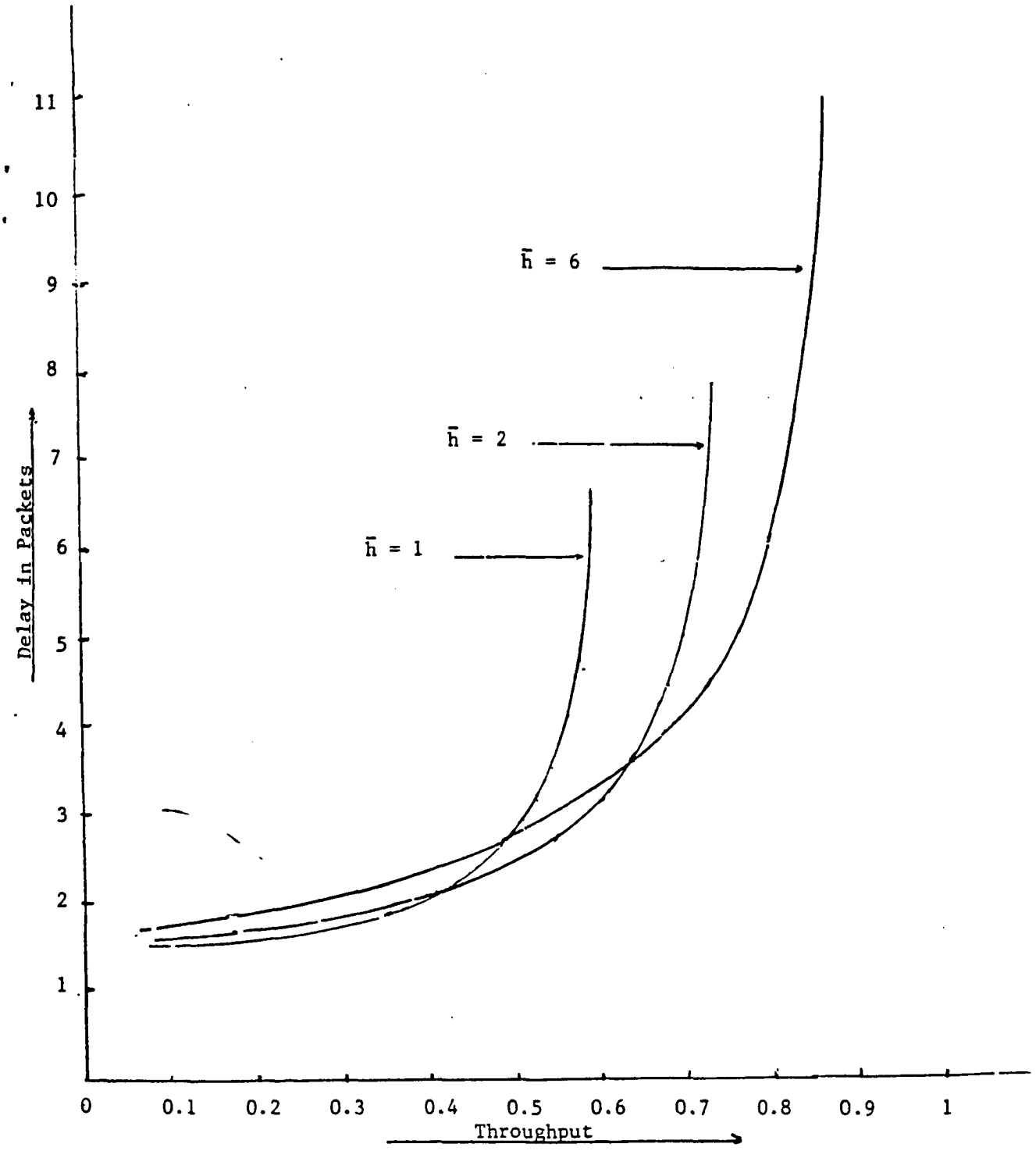
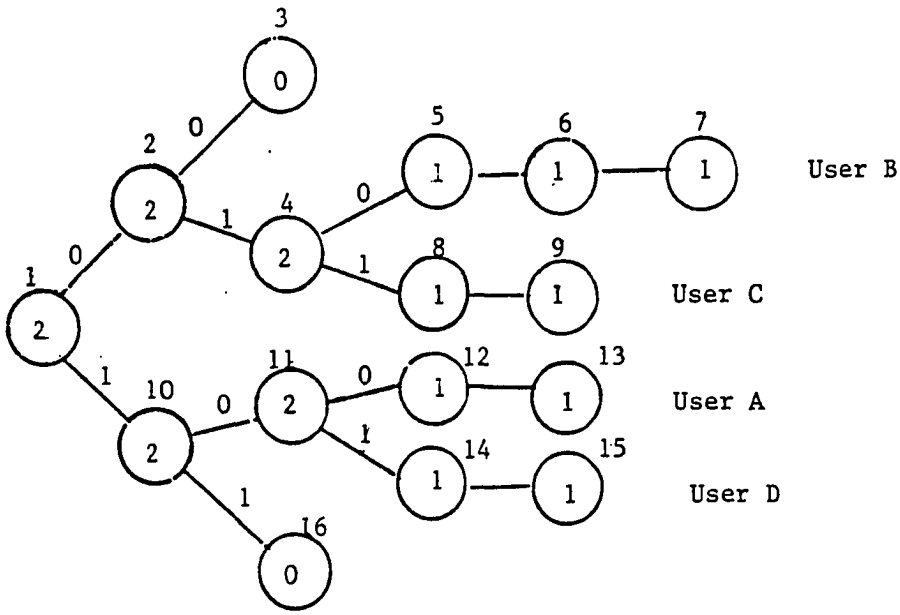


Fig. 2-4

3. Case of Infinite Number of Users

For infinite number of users, the algorithm is slightly different from the finite case. The basic algorithm can be described as follows; after a collision all users involved flip a binary fair coin, those flipping "0" retransmit in the next slot, those flipping "1" retransmit in the next slot after the collision (if any) among those flipping "0" has been resolved. No new packets may be transmitted until after the initial collision has been resolved completely. As in Section 2, for the case of message of h packets long, the first packet in the message is the one that goes through the collision. Once it succeeds in transmission it is followed by the rest of the message, (h-1) packets. The last packet in the message has a flag which indicates the end of the message.

An illustrative example is given below (see Fig. 3-1). We consider the users A,B,C and D having a message to transmit the number inside the circle indicates: 0 = empty slot; 1 = single packet; 2 = collision. After the collision in slot 1 users B and C flip "0" while both C and D flip "1" and after an empty slot B flips "0" and transmits its 3-packet message while C flips "1" and transmits its 2-packet message in slots 8 and 9. The CRI ends at the 16th slot.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A,B C,D	B,C		B,C	B	B	B	C	C	A,D	A,D	A	A	D	D	

CRI

Tree Algorithm with Variable Message Length and Variable Number of Users

Fig. 3-1

In the next section the delay analysis is carried out leading to the delay throughput characteristics. The analysis uses the results of J. Massey's [22] for infinite number of users tree algorithm.

3.1 Delay Analysis

Let the random variable Ya denote the length of the CRI in progress where a "random-chosen packet" arrives at its transmitter, then

$$E(Y_a) = E(Y_{ss}^2)/E(Y_{ss}) \tag{3.1}$$

where $E(Y_{ss})$ is the steady state length of the CRI in slots. Let the random variable Y_d denote the length of the CRI in which the randomly chosen packet departs from the system in the sense of being successfully transmitted (see fig. 3-2), and let X_d be the total number of packets in this CRI.

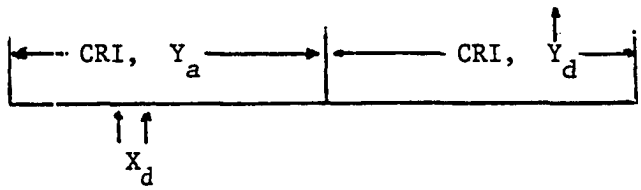


Fig. 3-2

As shown in Appendix II

$$E(Y_d) \leq a_u E(X_d) + 1 \tag{3.2}$$

which is a rather tight upper bound to $E(Y_d)$. where $a_u = 2.88 + \bar{h}-1$ and $\bar{h}$ the mean value of the message length in packets. Assuming Poisson arrivals, i.e., X_d is a Poisson random variable with mean λL, given $Y_a = L$

$$P(X_d = N/Y_a = L) = ((\lambda L)^N/N!)\, e^{-\lambda L} \qquad N = 0,1,2,\ldots \tag{3.3}$$

From (3.3) we imply that

$$E(X_d) = \lambda E(Y_a) \tag{3.4}$$

Replacing (3.4) into (3.2) we get

$$E(Y_d) \leq a_u \lambda E(Y_a) + 1 \tag{3.5}$$

We now introduce the random variable D representing the delay, in packets,

experienced by a randomly chosen packet, i.e., the difference between its arrival at the user's buffer and onset of its successful transmission. We note that a rather tight upper bound is

$$E(D) \leq 1/2\ E(Y_a) + E(Y_d - 1) \tag{3.6}$$

As follows from the facts (i) that on the average, the randomly chosen packet arrives at the midpoint of the CRI in progress, and (ii) that the latest of its successful transmission begins at the last slot of its departure CRI. Substituting (3.5) into (3.6) we get

$$E(D) \leq (1/2 + a_u\lambda)\ E(Y_a) \tag{3.7}$$

The random access system is said to be stable when $E(D)<\infty$ and from (3.7) follows that the algorithm is stable when $E(Y_a)<\infty$.

As shown in Appendix equation (I.5)

$$E(Y^2_{ss}) \leq a_u^2\ E(X^2_{ss}) - (4.369-2\bar{h})\ E(X_{ss}) + 1 \tag{3.8}$$

From the Poisson assumption eq. (3.3) implies that

$$E(X^2_{ss}) = E(X_{ss}) + \lambda^2 E(Y^2_{ss}) \text{ and } E(X_{ss}) = \lambda E(Y_{ss}) \tag{3.9}$$

Now replacing (3.9) into (3.8), dividing both sides by $E(Y_{ss})$ and using (3.1) we get

$$E(Y_a) \leq \frac{(-0.81+5.773\bar{h}+\bar{h}^2)\lambda+1/E(Yss)}{1 - a_u^2\ \lambda^2}$$

upper bounding $1/E(Y_{ss})$ by 1 we get

$$E(Y_a) \leq \frac{(-0.81 + 5.773\bar{h} + \bar{h}^2)\ \lambda+1}{1 - a_u^2\ \lambda^2} \tag{3.10}$$

The denominator of (3.10) must be positive in order to have $E(Y_a)<\infty$ or $1-a_u^2\ \lambda^2 > 0$ which gives the maximum allowable value of λ

$$\lambda Max = \frac{1}{a_u} = \frac{1}{1.8867 + \bar{h}} \tag{3.11}$$

and the corresponding maximum throughput is $\bar{h}\lambda_{max}$ packets/slot. Thus for stability we must have $\lambda<\lambda_{max}$. Now replacing (3.10) into (3.7) We get $\{(1/2) + a_u\lambda\}\ E(Y_a)$ which is a rather tight upper bound of the total packet delay $E(D)$..

In Fig. 3.3 we plot the upper bound of the average packet delay in slots versus the channel throughput $\bar{h}/\lambda$. When $\bar{h} = 1$, we get as maximum throughput the Capetanakis-Massey result of 0.345. For $\bar{h} = 2$ the maximum throughput is 0.514 and for $\bar{h} = 8$ the maximum throughput becomes 0.809, and as the average message length increases further the maximum throughput approaches one.

In the following section we consider another version of the above tree algorithm with variable message length which results in higher maximum throughput. We call it Reservation Tree Algorithm.

Tree Algorithm with variable message length (infinite population of users)

$E(D)$ V_s $\bar{h}\lambda$

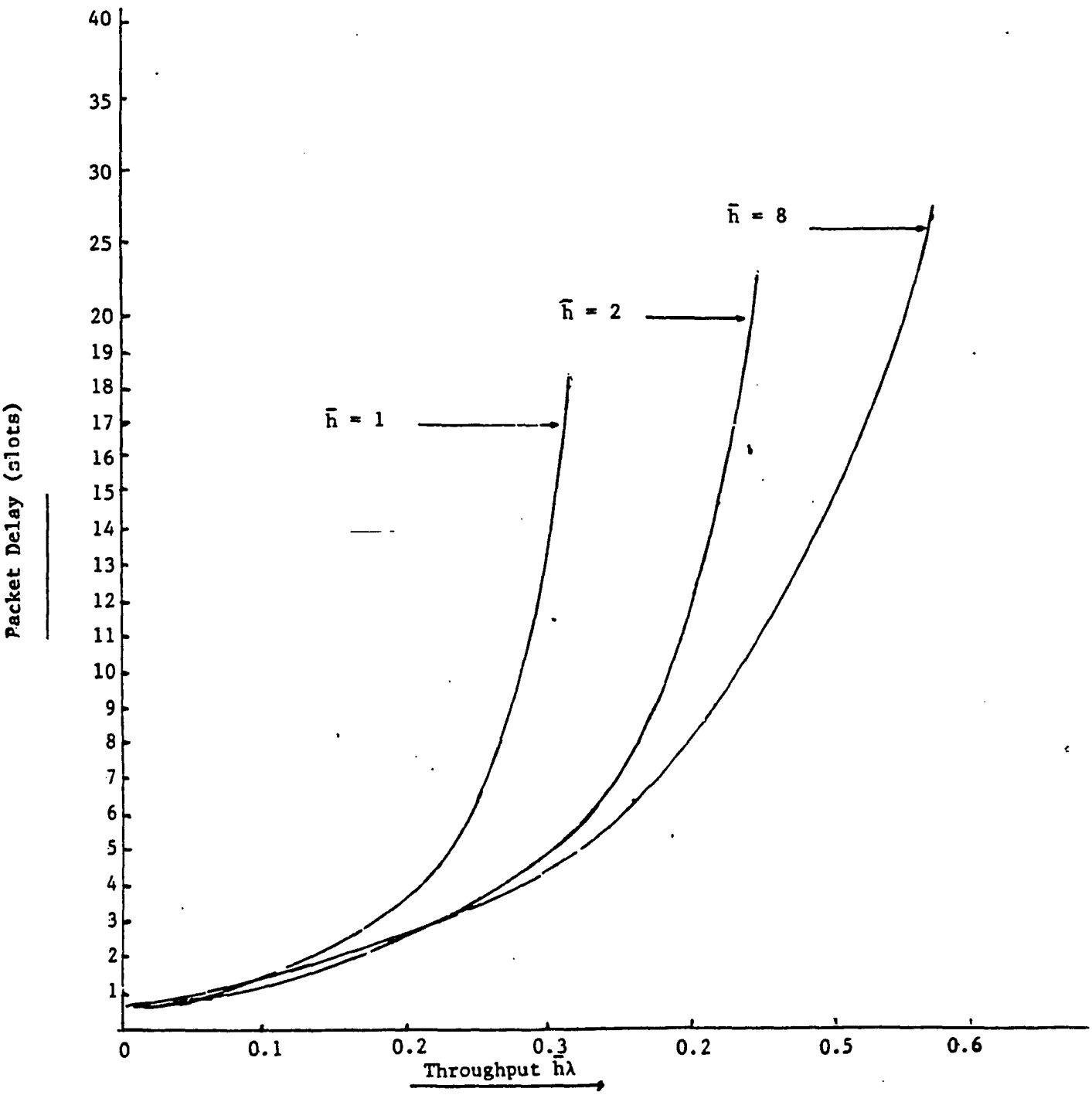


Fig. 3-3

4. The Reservation Tree Algorithm

The R-Tree Algorithm is similar to the one considered in the previous section. The difference is that instead of transmitting the first packet of the message to resolve the collision among the users, "test" packets are used. The test packet is much smaller than a data packet. If each test packet has length t and the data packet has length T, then $\tau = t/T$ having typical values of 0.1 or 0.2. The test packet carries only feedback information about the length of the message the user has to transmit.

More specifically the algorithm works as follows: Each user that has a message to transmit, first transmits its test packet. The collision between the test packet will be resolved, according to Tree Algorithm, during a CRI named "test CRI." At the end of the test CRI each user has all the feedback information and it knows where to transmit its message without conflict and how many slots in the channels are reserved for it. This interval that the users transmit their messages without conflict is named "Reservation Interval" (see Fig. 4.1).

In the analysis we consider infinite population of users and it is similar to the analysis of the previous section.

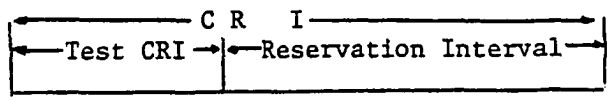


Fig. 4-1

4.1 Delay Analysis

For R-Tree Algorithm the conditional CRI length has two parts, the Test CRI, due to the collision resolution among the test packets of length bounded by $(2.886N+1)\tau$, and the Reservation Interval of length $\bar{h}N$ where N is the number of users collided in the first slot of the CRI.

Thus $L_N \leq (2.886N+1)\tau + \bar{h}N = (2.886\tau + \bar{h})N + \tau$ multiplying both sides by $P(X_{ss} = N)$ and summing over N we get $E(Y_d) \leq a_u E(X_d) + \tau$ where $a_u \triangleq (2.886\tau + \bar{h})$. By the Poisson assumption we have $E(X_d) = \lambda E(Y_a)$, replacing it into the above equation we get

$$E(Y_d) \leq a_u \lambda E(Y_a) + \tau \tag{4.1}$$

The second moment of the conditional CRI length is derived similarly as in Appendix I. Here we have

$$E(Y_{ss}^2) \leq a_u^2 E(X_{ss}^2) + (2\tau\bar{h} - 2.336\tau^2)E(X_{SS}) + \tau^2 \tag{4.2}$$

replacing equation (3.9) into (4.2) dividing both sides by $E(Y_{ss})$ and using (3.1) we get

$$E(Y_a) \leq \frac{(6\tau^2 + 9.773\tau\bar{h} + \bar{h}^2)\lambda + \tau^2}{1 - \lambda^2(a_u^2)} \tag{4.3}$$

The upper bound to the total delay is again given by (3.6), i.e.,

$$E(D) \leq 1/2E(Y_a) + E(Y_a - 1)$$

replacing $E(Y_a)$ by (4.1) we get

$$E(D) \leq (1/2 + a_u\lambda)E(Y_a) + \tau - 1 \tag{4.4}$$

where $E(Y_a)$ is given in equation (4.3).

For stability we must have $E(D) < \infty$ or $E(Y_a) < \infty$ or $1 - \lambda^2 a_u^2 > 0$ which gives the maximum value of λ for a stable system

$$\lambda max = \frac{1}{a_u} = \frac{1}{2.886\tau + \bar{h}} \tag{4.5}$$

and the corresponding value of the maximum throughput is $\bar{h} \cdot \lambda max$.

In Table 4-1 we have the maximum throughput for different values of τ and $\bar{h}$. For $\bar{h} = 1$ we see the tremendous increase of the maximum throughput 0.776 for $\tau = 0.1$ and 0.634 for $\tau = 0.2$, compared with the 0.345 of single packet message (Capetanakis-Massey).

In Fig. 4-2 the delay throughput characteristics are plotted for $\tau = 0.1$

	$\tau = 0.1$	$\tau = 0.2$
$\bar{h}=1$	0.776	0.634
$\bar{h}=2$	0.873	0.776
$\bar{h}=8$	0.965	0.939

Table 4-1

Delay vs Throughput Characteristics

R-Tree Algorithm (infinite population of users)

$\tau = 0.1$

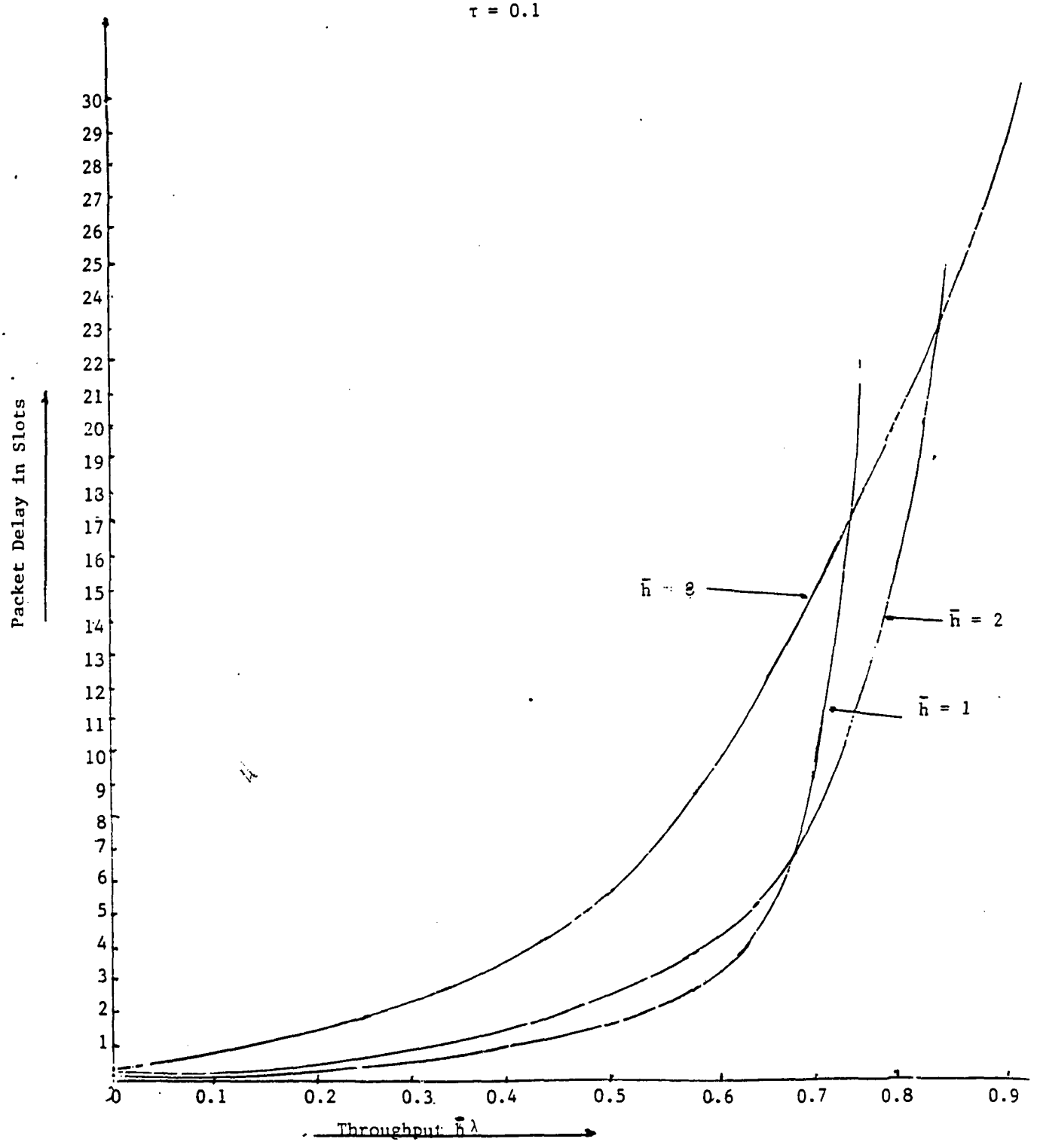


Fig. 4-9

5. The Alternating Tree Algorithms

In the algorithms discussed so far we consider zero propagation delay. In this section the problem of long propagation delay is addressed.The proposed scheme named Alternating Tree Algorithms is designed to make use of the channel time at long propagation delays (0.25 sec for satellite channels).

More specifically the A-Tree Algorithms are k-interleaving Tree Algorithms, i.e., between the slots of the ith Algorithm are k-1 slots of the other k-1 algorithms (see Fig. 5-1). The number of the algorithms is chosen large enough to overcome the round trip propagation delay.

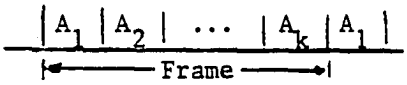


Fig. 5-1

The CRI's of each of the k-algorithms start, in general, in different frames. The interval between two successive starting points of CRI's is called Arrival Iinterval (A-I). The packets that arrived in each A-I will resolve

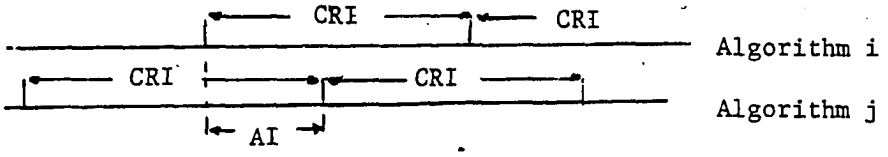


Fig. 5-2

their collisions in the next starting CRI, i.e., in CRI 2 of the algorithm j for the example of Fig. 5-2. In other words, an arrived packet is going to that algorithm whose CRI immediately starts next after its arrival. This means no user is engaged with a specific algorithm

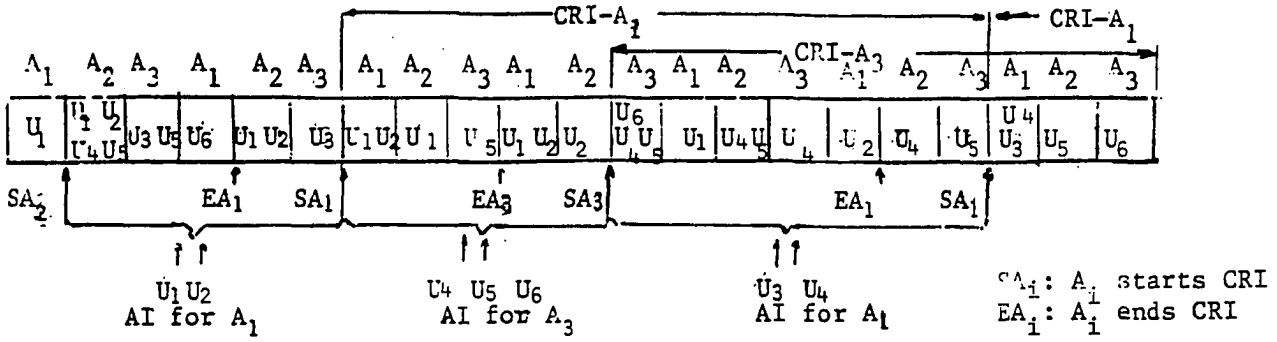


Fig. 5-3

An illustrative example of the A-Tree Algorithms is given (see Fig. 5-3). We considered k=3 interleaving algorithms. The packets that arrived between the successive starting points of A_2 and A_3 (U_1 and U_2) will resolve their collision in the CRI of A_1 while the packets that arrived in the A-I for A_3 (U_4, U_5, and U_6) will resolve their collision in the CRI of A_3, etc.

5.1 The Average Packet Delay

We consider as Interleaving Tree Algorithm the one with infinite population of users. The messages that arrived in an Arrival Interval will resolve their collision in the following CRI whose steady state length is Y_{ss} that is measured in frames where a frame is k slots long. From the Poisson assumption we have $E(Y_{ss}) = 1/\lambda E(X_{ss})$ where X_{ss} the number of messages arrived in the A-I or the number of users colliding in the following CRI. This means that the A-I is Y_{ss} slots long in steady state.

The upper bound of the total delay D is

$$E(D) \leq 1/2E(Y_a) + E(Y_d' - 1) \tag{5.1}$$

where Y_a is the Arrival Interval where a randomly chosen packed arrived

$E(Y_a) = E(Y_{ss}^2)/E(Y_{ss})$ and Y_d' the length of the CRI in frames, which implies $E(Y_d' - 1)$ frames is $kE(Y_d)-k$ slots. But $E(Y_d) \leq a_u \lambda E(Y_a) + 1$ as implied from (3.5). Replacing it into (5.1) we get the total delay in slots.

$$E(D) \leq (1/2 + k\lambda a_u) E(Y_a) \tag{5.2}$$

where $a_u = 2.8867 + \bar{h} - 1$ and $\bar{h}$ is the average length, $E(Y_a)$ is given by the same expression as in equation (3.10). Thus the maximum throughput of the A-Tree Algorithm is the same as the maximum throughput of each of the interleaving algorithms, that is, $\bar{h}/a_u$ in our case, while the delay is increased by $(k-1)\lambda a_u E(Y_a)$ slots.

The advantage of the above scheme is that it can be used in a large or small population of users, while it succeds to overcome the round trip delay with the only increace on delay in the departure CRI. Also the behavior of the a A-Tree Algorithms and each of the interleaving is the same.

6. The Hybrid TDMA-Tree Algorithm for Multibeam Satellites

Here we consider the satellite switched multibeam system, with n zones and with each zone covering a large population of spacially isolated users. The proposed scheme is a combination of the Time Division Multiple Access (TDMA) and the Tree algorithm with variable message length proposed in section 3 and is adapted for multibeam systems. More specifically the scheme has as follows. Within each zone the number of users are divided into n groups according to their destination. For each destination group is dedicated one slot in the TDMA frame. This slot is used by the in the group to resolve their collision (in successive frames) to resolve their collision according to the tree algorithm. The TDMA frame, which is n slots long, is assumed long enough to overcome the round trip delay.

The switching is considered to be solved with the switch following a predetermined sequence, which means that while the TDMA of source zone 1 have slots assigned to destination zones in the sequence 1,2,...,n ; the assignment of slots on the ith source zone is cycly permutated i.e. in the sequence i,i+1,...,n,1,...i-1 .Thus while the ith source tranmits to the jth destination zone the i+1 will be tranmitting to the j+1 destination zone etc. The above is illustrated in Fig 6.1

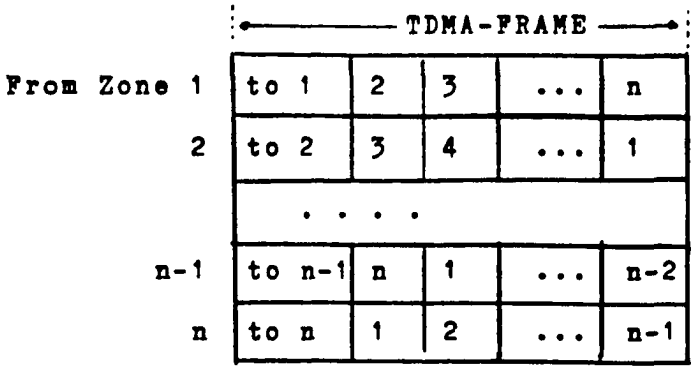


Fig 6.1

The delay analysis is based on the analysis of tree algorithm with large population of users given in section 3. Now if λ the message arrival rate per slot per zone then the message arrival rate per slot per zone per destination will be λ/n

Since we assume poisson arrival

$$P(X_d=N/Y_a=nL)=[(\lambda L/N!)^N]e^{-(\lambda/n)nL}=[(\lambda L)^N/N!]e^{-\lambda L} \quad (6.1)$$

The message arrival CRI Y_a is L frames or nL slots and X_d is as in section 3.1. From (6.1) we imply that $E(X_d)=\lambda \cdot E(Y_a)$, replacing it into (3.2) we get

$$E(Y_d) \leq a_u \lambda E(Y_a)+1 \quad (6.2)$$

With $a_u=2.88+\tilde{h}-1$, and $E(Y_a)$ is in frames.

Replacing (6.2) into (3.6) we get the upper bound of the total delay in slots

$$\overline{E(D)} \leq n(1/2+a_u\lambda)E(Y_a) \quad (6.3)$$

Where $E(Y_a)$ is given in (3.10) and the maximum throughput is the same as in (3.11)

Thus from (6.3) follows that the delay vs throughput characteristics are the same as in Fig 3.3 but with the packet delay instead of slots in frames.

Comparing the hybrid-TDMA with the Random TDMA given in [5] the first has the advantage that carries small delays is stable and does't require on board buffering.

CONCLUSION

In this thesis we have studied the multiple access and the switching problems for multibeam satellites, and we proposed several algorithmic solutions. In chapter 1 we presented and evaluated the algorithms SA1 and SA2 for SS/TDMA systems. We have shown that both of them are optimal,i.e achieve the minimum schedule length. Simulation results show that the number of generated switching, for both of them, are much lower than the upper bound n^2-2n+2. Further more the overall complexity has been reduced to the order of $O(kn^3)$, where k is much smaller than the number of switching. The above is achieved with the aid of latin squares for SA1 and submatrices or blocks for SA2.

In chapter 2 we have studied the problem of mimimizing the number of switchings in an SS/TDMA system in such a way that also continuous transmission of the message is permitted. Firstly using the method of superimposing a Latin Square (LS) on the traffic matrix T and trasmitting T according to LS, we have developed an algorithm which achieves the minimumm possible schedule length, while the number of mode matrices is bounded by n (n: size of the traffic matrix). This algorithm, named MSL algorithm, has computational complexity bounded by $O(n^3)$. Along with the above method we also use a suboptimum algorithm to compare with.

Secondly, the method of nonconflicting blocks is utilized as an extension of the first method. We have developed a

heuristic algorithm to implement this method.

Simulations have been carried out to compare the generated schedule lenghth of each algorithm with the critical line sum of the traffic matrix and with each other. The simulation results show that the method of nonconflicting blocks is the best for a 4x4 traffic matrix although the algorithm used is heuristic. The MSL algorithm generates 5 to 10% average percentage overlenghth of the of the average critical line sum .

In chapter 3, we presented and analyzed a class of the tree algorithms with variable message length. All the presented algorithms show considerable impovement on the maximum throughput, which increases as the message length increases i.e.for h=8 the maximum throughput of infinite population algorithm is 0.809. Further more, the above algorithms have all the good properties of the tree algorithm, i.e are stable and fair for $\lambda < \lambda_{max}$. The reservation tree algorithm presented in in section 4, shows even more on the maximum throughput. Comparing the above algorithms with the R-ALOHA presented in [35], we see that they dont have the disadvantage of the Aloha protocol, i.e. instability. Also the TDMA frame in R-Aloha doesn't permit a large increase to the population of users. In section 5, we presented the Alternating Tree Algorithms that adapts the satellite's channel long propagation delays. The advantage of the above scheme is that adjusts itself to the large or small population of users while the delay increase is only on the

departure CRI as shown in equation (5.2).

Finally we presented and analyze the hybrid TDMA-Tree algorithm for multibeam satellites, which can be used with large number of users and does not require on board buffering.

APPENDIX I

THE MAX-MIN ALGORITHM

The max-min algorithm finds an SDR of maximum cardinality which also maximizes the smallest element in it; i.e. given an nxn QDS matrix the max-min algorithm will select, among the existing systems of n distinct representatives, the one which maximizes the smallest element in it.

The problem of finding an SDR in a matrix $D=\lceil d_{ij} \rceil$ is eqiuvalent to the graph theory problem of finding a maximum cardinality matching on a bipartite graph $G=\{S,T,A\}$; where $S=\{1,2,..,n\}$ and $T=\{1,2,..,n\}$. For the case of weighted bipartite graph, $A=\{(i,j)/d_{ij} \geq 0\}$. The max-min matched bipartite graph $G_m=\{S,T,A_m\}$ has the minimum weight of an edge maximized, i.e. $A=\{(i,j)/d_{ij} \geq W\}$ where W is a threshold. The max-min bipartite matching is given in the book "Combinatorial Optimization: Networks and Matroids" by E.L.Lawler [24]. (Threshold method, page 198). The complexity of the above algorithm is polynomial bounded to n^3; $O(n^3)$.

Thus we can use this algorithm described in [24] to find an SDR to our traffic matrix $D=\lceil d_{ij} \rceil$ by constructing a bipartite graph $G=\{S,T,A\}$, which has as adjacency matrix the traffic matrix $D=\lceil d_{ij} \rceil$ with $i \in S$ and $j \in T$. Then the max-min matched bipartite graph corresponds to the max-min SDR in D.

The above algorithm is also "translated" for matrix form data structure. Starting with an empty SDR and a large "threshold" W, at any given step the max-min SDR of

cardidality Y has been obtained. The algorithm tries to find an augmenting set in the subset of entries d_{ij} such that $d_{ij} \geq W$

(An augmenting set is defined in step 2). If the augmenting set is found then the max-min SDR of cardinality Y+1 results. If not, the threshold is reduced so that augmentation is possible. The Algorithm stops when no augmentation is found, regardlees of W.

In the following algorithm a line index is said to be **Exposed** if the line to which it belongs has no represenative in the current solution. X is the set of chosen representatives, W and π_j. $1 \leq j \leq n$ are the weights used in the algorithm to select the representatives.

STEP 0: The matrix $D=[d_{ij}]$ is given, $0 \leq i,j \leq n$. Set $X=0, W=+\infty$ and $\pi_j = -\infty$ for each j. No row index or column index is labeled.

STEP 1: a) Give the label $\emptyset$ to each exposed row index.

b) If there are no unscanned labels, but each unscanned label is on column index j for which $\pi_j < W$, then $W=\max\{\pi_j / \pi_j < W\}$

c)Find an index $\bar{h}$ with an unscanned label where either $\bar{h}$ is a row or else $\bar{h}$ is a column index and $\pi_{\bar{h}} \geq W$. If $\bar{h}$ is a row index go to step 1d); If $\bar{h}$ is a column index go to step 1e).

d)Scan the label on row index $\bar{h}$ as follows. For each $d_{\bar{h}j}$ not in X, if $\pi_j \leq d_{hj}$ and $\pi_j < W$, the give the column index j the label $\bar{h}$ (replacing any existing label) and $\pi_j = d_{\bar{h}j}$. Return to step 1b).

e)Scan the label on column index $\bar{h}$ as follows. If index $\bar{h}$

is exposed,go to step 2). Otherwise, identify the unique row index i such that $d_{i\bar{h}}$ in X and give the row index i the label $\bar{h}$.Return to 1b)

STEP 2: An augmenting set has been found. The elements in the set are identify as follows. Starting from the column index $\bar{h}$ (identified in step 1e) first form the following sequence by "backtracking" from label to label:

$h_p h_{p-1} h_{p-2} \cdots h_0$ Where $h_p = \bar{h}$

$h_{p-1} = \text{label}(h)$

$h_{p-2} = \text{label}(\text{label}(h))$

. . .

The sequence stops at h_0,that has the label 0. Now identify the following set of entries in D, which is the augmenthing set:

$$S = \{d_{h_p,h_{p-1}}, d_{h_{p-1},h_{p-2}}, \ldots, d_{h_1,h_0}\}$$

Augment S by adding to X all those which are in S. Remove all the labels from the indexes. Set $\pi_j = -\infty$ for all J's. Return to step 1a)

STEP 3: No augmenting set exists, and X is a max-min system of distinct representatives of maximum cardinality.

APPENDIX II

THE CONDITIONAL CRI LENGTH

Let $L_N = E(Y_{ss}/X_{ss}=N)$ be the conditional CRI length given N users contending for transmission and let $S_N = E(Y^2_{ss}/X_{ss}=N)$ be the conditional second moment of the CRI length, then the conditional variance V_N is given by $V_N = S_N-(L_N)^2$ If we consider large number of users N=2n collided at the first slot of the CRI, in the following slot very close to half of the users will flip 0 and half will flip 1 (see fig.II-1).

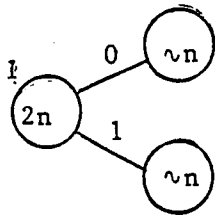


Fig.II-1

Thus the CRI length $L_{2n} \simeq 1+2L_n$ $n>>1$. The solution of this recursive equation is $L_N = \alpha N-1$ which is very accurate even for small values of N. Similarly for the variance we conclude that $V_n = \beta N$ $N>>1$

In [3,2] the constant α is calculated to be $2.8810 \leq \alpha < 2.8867$ while $3.359N \leq \beta < 3.404$

Thus the upper bound of the conditional CRI length for the variable message algorithm is given below. Where the traffic-independent property of the Tree algorithm is used, i.e., given N, L_N depends only on the results of the coin tosses performed internally in the algorithm, and hence is independent of the traffic statistics that led to the given value N.

$$L_N \leq 2.8867N + (\bar{h}-1)N + \delta_{0N} - 1.886\delta_{1N} \quad N \geq 0$$

where $(\bar{h}-1)N$ represents the CRI part due to transmission of the rest of the message for each user. The terms $\delta_0 N$ and $\delta_1 N$ include the case of small values of N. (N=0,1) with $\delta_{iN}=1$ for i=N and 0 for $i \neq N$. For simplicity we bound those terms by 1. Then we can write

$$L_N \leq a_u N+1 \quad (1.1)$$

where $a_u = 2.8867 + \bar{h}-1$ and $\bar{h}$ the average message length.

The second moment of the conditional CRI length is $S_N = V_N + (L_N)^2$ is upper bounded by

$$\alpha^2N^2 - (2\alpha-\beta)N+1 = 8.333N^2-2.369N+1 \text{ for } N\geq 3 \quad (I.2)$$

Thus $$S_N = E(Y_{ss}^2/X_{ss}=N) = E\{[Y+(\bar{h}-1)N]^2/X_{ss}=N\} \quad (I.3)$$

where Y is the CRI length without including the rest of the message (h-1), and is bounded by the above expression (I.2). Now expanding (I.3) and replacing in it (I.1) and (I.2) we get

$$S_N \leq a_u^2N^2 - (4.369-2\bar{h})N+1 \quad (I.4)$$

Multiplying both sides of the above by $P(X_{ss}=N)$ and summing over N we get

$$E(Y_{ss}^2) \leq a_u^2 E(X_{ss}^2) - (4.369-2\bar{h})E(X_{ss}) + 1 \quad (I.5)$$

where $$E(Y_{ss}^2) = \sum_{n=o}^{\infty} S_N P(X_{ss}=N) \quad E(X_{ss}) = \sum_{n=o}^{\infty} NP(X_{ss}=N)$$

and $$E(X_{ss}^2) = \sum_{n=o}^{\infty} N^2 P(X_{ss}=N)$$

REFERENCES

[1] Jin-Fu Chang,"A multibeam packet satellite using random access technics" IEEE Trans. on Com.,vol. com-31,October 1983

[2] J.K.DeRosa and L.H.Ozarow,"Packet switching in a processing satellite" Proc. IEEE ,vol.66,pp 100-102, Jan. 1978

[3] R.E.Evans,"ALOHA/TDM systems with multiple downlink capacity",IEEE Trans. on com.,com-27,March 1979

[4] J. Gadre and T.E. Stern,"A comparison of multiple access protocols of packet switching in satellite switched multibeam systems" ICC conf.proc.,pp5341-6, June 1980

[5] M.Kawai,T.N.Saadawi,D.L.Schilling, "Random TDMA access protocol with applications to multibeam satellites" ICC proc 1981

[6] T.Scarcella and R.V. Abbot "Orbital efficiency through Satellite Digital Switching" IEEE communication magazine May 1983 Vol.21,No.3

[7] A.S.Acampora and B.R.Davis,"Efficient utilization of satellite tansponders via Time-Division multibeam scanning", Bell system technical jurnal,vol.57,pp2901-14,October 1978

[8] A.S.Acampora,C.Dragone,D.O.Reudink,"A satellite system with limited scan spot beams",IEEE Trans. on com ,vol. com-27,October 1979

[9] A.J.Frank,T.E.Stern,"On board demand scheduling of a

SS/TDMA multibeam satellite with interated circuit and packet-switching" 9th conf. on digital satellites 1983

[10] I.Gopal,M.Bonucelli and C.K.Wong,"Scheduling in multibeam satellites with interfering zones" IBM Research Report RC9663(#41124) April 1982

[11] Y.Ito,Y.Urano,T.Maratani, "Analysis of a switch matrix for SS/TDMA system" Proc. IEEE vol.65 pp 411-419,Mar 1977

[12] T.Inukai, "Comments on analysis of a switch matrix for an SS/TDMA system" Proc. IEEE vol.66 pp 1669-1670,Dec 1978

[13] T.Inukai,"An efficient SS/TDMA time slot assignment algorithm"IEEE Trans. comm. vol.com-27,pp1449-1455,Oct1979

[14] T.E. Stern,"packet scheduling protocols for a multiple beams comm satellites" in proc.Int.Symp.Information Theory,Italy,pp130-131,June1979

[15] W.W. Wu,"On the efficiency of traffic assignment in SS/TDMA system" Presented on the 4th conf. Digital Satellite communications, Montreal, Canada, Oct1978.

[16] G.Bongiovanni,D.Coppersmith and C.K.Wong,"An optimal time slot assignment algorithm for an SS/TDMA system with variable number variable number of transponders" IEEE Trans.Comm.vol.com-29,pp721-726,May 1981

[17] I.S. Gopal,D.coppersmith,and C.K.Wong,"Minimizing packet waiting time in a multibeam satellite system",IEEE Trans. comm. vol.comm-30,pp305-316 Feb 1982

[18] G.Bongiovanni,D.T.Tang and C.K.Wong,"A general multibeam satellite switching algorithm ",IEEE Trans.Comm.vol.com-29,pp1025-1036 July 1981

[19] I.L. Lewandowski,J.W.S.Liu and C.L.Liu "SS/TDMA Time slot assignment with restriG.Maral,M.Bousquet,P.Wattier,"A practical approach to SS/TDMA time slot assignment ICC conf 1983

[21] I.Gopal and C.K.Wong,"Minimizing the number of switching in an SS/TDMA system " IBM Research Report RC9738(#42886) November 1982

[22] K.S.Natarajan and S.B.Calo,"Time slot assignment of an SS/TDMA system with minimum mumber of switchings"IBM Research Report 1981

[23] H.J. Ryser, Combinatorial Mathematics,Math Asso. of America 1963

[24] E.L. Lawler,Combinatorial Optimization: Networks and Matroids. New York: Holt Rinethart and Winston 1967

[25] C.Papadimitriou,K.Steiglitz Combinatorial Optimization Prentice Hall 1982

[26] J.E.Hopcroft and R.M.Karp,"An $n^{5/2}$ algorithm for maximum matching in bipartite graphs",SIAM J.Computing,vol-2 pp225-231,Dec 1973.

[27] J.Denes and A.D.Keedwell,Latin Squares and their applications.New York: Academic 1974

[28] M.R.Garey and D.S.Johnson;Computers and Intractibility; W.H.Freeman and Co.,San Francisco 1979

[29] S.Even,A.Itai and A.Shamir,"On the complexity of the Time Table and Multicomodity Flow problems",SIAM J. on computing,vol.5,pp691-703 1976

[30] S.Lam,"Satellite packet communications multiple access

protocols and performance" IEEE Trans. Com.,vol.com-27 October 1979

[31] John Capetanakis,"Generalized TDMA:The multi-accessing Tree protocol",IEEE Trans. Com.,vol.com-27,October 1979

[32] James Massey,"Collision Resolution Algorithms and Random-Access communications",School of Engineering and Applied Science,UCLA California 90024

[33] T.N.Saadawi and A.Ephremides,"Analysis of the tree algorithm with a finite number of users" ICC conf. Dec. 1981

[34] T.T.Liu and D.Towsley,"Window and Tree protocols for satellite channels", INFOCOM April 1983

[35] S.Lam,"Packet broadcast networks-A performa analysis of the R-Aloha protocol",IEEE Trans. Computers,vol.C-29,July 1980

[36] F.Borgonovo and L.Fratta,"A collision resolution algorithm for random access with echo" ICC conf. 1983

[37] R.Cruz and Hujek,"A new upper bound to the throuput of the multiaccess broadcast channel" IEEE Trns. on Inf. Theory,May 1982.