

# Efficient Time Slot Assignments for TDM Multicast Switching Systems

Kwan L. Yeung K. F. Au-Yeung Li Ping

Department of Electronic Engineering

City University of Hong Kong

Tat Chee Avenue, Hong Kong

kyeung@ee.cityu.edu.hk

**Abstract**— This paper focuses on designing efficient multicast time slot assignment (MTSA) algorithms for TDM switching systems. Based on a *packet compatibility matrix*, the MTSA can be transformed to the well-known graph-coloring problem. We thus show that the MTSA problem is NP-complete. A lower bound on the frame length of a multicast time slot assignment is then found to be the clique number of the MTSA equivalent graph. Two efficient MTSA algorithms, called the contention-based ordering (CBO) algorithm and the hybrid ordering algorithm, are proposed. Their performance is compared with the three existing algorithms and the lower bound through extensive simulations. We found that the CBO algorithm, which has one of the lowest computational complexities, gives the best performance among all five algorithms studied. The average frame length generated by the CBO algorithm is within 1% above the lower bound. We also show that (i) the previously reported DAC algorithm has the poorest performance despite its highest complexity, and (ii) the previously reported CCBO algorithm has only a comparable performance to the simple greedy algorithm.

## 1 Introduction

Time-division multiplex (TDM) switching has widely been employed in terrestrial and satellite communication networks to concentrate traffic from low bandwidth sources onto high bandwidth lines. An  $N \times N$  TDM switch is shown in Fig. 1. It routes time-multiplexed traffic from its inputs to its outputs. Switching operation is made up of frames and each frame is divided into time slots. Each time slot can accommodate one packet. A switch configuration is an interconnection pattern of the switch such that at most one packet can be transmitted by an input and one packet can be received by an output. A switching conflict occurs if two or more packets are transmitted to the same output in the same time slot.

The time slot assignment (TSA) problem in a TDM switching system is to find a conflict-free assignment of packets to slots such that the frame length, i.e. the number of time slots required for switching the traffic from switch inputs to outputs, is minimized. For a given traffic matrix (to be defined), an optimal TSA is an assignment that has the minimum frame length over all possible conflict-free assignments. For unicast TDM switching systems, many optimal algorithms with polynomial time complexity have been proposed [1, 2, 3, 4]. Most of them use a maximum cardinality algorithm known as sys-

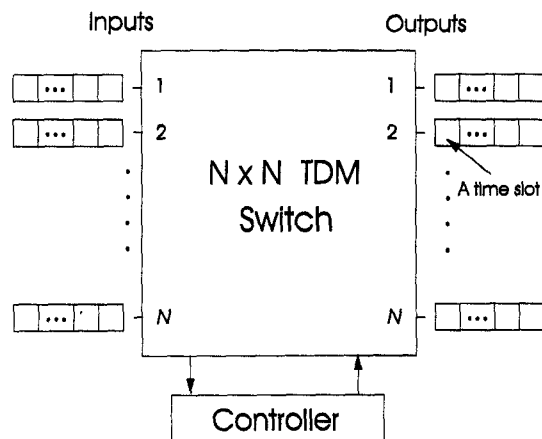


Figure 1: An  $N \times N$  TDM switch.

tem of distinct representatives (SDR) [1], which alone has a time complexity of  $O(N^4)$  [5].

A multicast switch is a switch which can connect any input port to any subsets of its output ports. The TSA problem for a multicast TDM switch, or MTSA problem, is also to find a conflict-free assignment of packets to slots such that the frame length is minimized. One way to solve the MTSA problem is to transform it to a unicast TSA problem by splitting all multicast packets into unicast packets. Then the traditional unicast algorithms can be applied. However, call (or packet) splitting would add extra complexity to the switch and in some applications (e.g. distributed data processing) all copies of a multicast packet are required to reach their respective destinations in the same time slot. If no call splitting is allowed, it has been shown [6] that the MTSA problem is NP-complete and we have to rely on heuristic algorithms for sub-optimal performance.

In this paper, two efficient multicast time slot assignment algorithms, called contention-based ordering (CBO) algorithm and hybrid ordering (HO) algorithm, are proposed. In the next section, the multicast time slot assignment problem is formulated. Using a *packet compatibility matrix*, we show that the MTSA problem is equivalent to the well-known graph-coloring problem. A lower bound on the length of the time slot assignment is thus found to be the clique number of the MTSA equivalent graph. In Sections 3 and 4, the CBO algorithm and the hybrid ordering algorithm are presented. In Section 5, their performances are compared with the lower

$$A = \begin{array}{c} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \end{array} = \begin{array}{c|cccc|cccc} & \text{Source Part} & & & & \text{Destination Part} & & & & \\ \hline & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Figure 2: A traffic matrix  $A$  for a  $5 \times 5$  multicast switch.

$$C = \begin{bmatrix} - & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & - & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & - & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & - & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & - & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & - & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & - & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & - & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & - & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & - \end{bmatrix}$$

Figure 3: A packet compatibility matrix derived from traffic matrix shown in Fig. 2.

bound and three existing algorithms in the literature. We show that the average frame length generated by the CBO algorithm is the shortest and is within 1% above the lower bound. In Section 6, we conclude the paper by highlighting the main contributions of this paper.

## 2 Problem Formulation

Let each packet at the inputs of an  $N \times N$  switch be represented by a *traffic vector* [6]  $V = [v_1, v_2, \dots, v_{2N}]$ . A traffic vector consists of two parts with exactly one 1 in the first  $N$  entries (source part) and at least one 1 in the subsequent  $N$  entries (destination part). If  $v_i = 1$  and  $i \leq N$ , the packet is at input  $i$  and its destinations are the set of outputs with  $v_j = 1$  and  $j > N$ . Throughout the paper, the term packet and the traffic vector will be used interchangeably.

Let an  $\alpha \times 2N$  *traffic matrix*  $A = [a_{ij}]$  denote the collection of  $\alpha$  traffic vectors at the switch inputs (Fig. 2), where each row of  $A$  corresponds to a traffic vector. Further let a *transmission matrix*  $T = [t_{ij}]$  denote the set of traffic vectors that can be transmitted in the same time slot (without conflict).  $T$  is a matrix that has at most one 1 in each column.

Let an  $\alpha \times \alpha$  *packet compatibility matrix*<sup>1</sup>  $C = [c_{ij}]$  denote

<sup>1</sup>In cellular mobile radio systems, a channel compatibility matrix [7, 8] is usually used to represent the minimum channel separation between channels assigned to two different cells.

if any two packets in a traffic matrix can be transmitted in the same time slot without conflict:  $c_{ij} = 1$  if packets  $V_i$  and  $V_j$  can be transmitted in the same time slot and  $c_{ij} = 0$ , otherwise. The compatibility matrix is symmetrical about the diagonal and all entries on the diagonal are "don't care" terms and denote by "-". An example is shown in Fig. 3.

**Theorem 1** *MTSA problem is NP-complete.*

**Proof:** From compatibility matrix  $C$ , a graph can be constructed by representing each traffic vector by a vertex, and two vertices are connected by an edge if  $c_{ij} = 1$ . Then assigning packets to slots for conflict-free transmission is the same as coloring the graph with different colors for adjacent vertices. Thus MTSA becomes equivalent to the conventional *graph-coloring problem*. The graph-coloring problem has been proved to be NP-complete [9]. The MTSA problem we considered here is therefore also NP-complete.

A *clique* in a graph  $G$  is a subgraph of  $G$  that is a complete graph. The *clique number* of graph  $G$ , denoted by  $\beta$ , is the largest integer such that  $G$  contains a clique of  $\beta$  vertices.

**Theorem 2** *The clique number  $\beta$  of the graph constructed from the compatibility matrix is a lower bound on the frame length for a MTSA problem.*

The proof of the above theorem is trivial. The problem for finding the clique number for a graph is not NP-complete if the vertex degree is bounded by a constant value [9]. For the MTSA problem, the bound on the vertex degree is the number of traffic vectors  $\alpha$ . In Section 5, the performances of various MTSA algorithms are compared with this lower bound.

## 3 Contention-Based Ordering Algorithm

Let  $R_i$  be the number of 1 entries in row  $i$  of a compatibility matrix. Let  $R_p = \max_i \{R_i\}$ . Then packet  $V_p$  cannot be transmitted in the same time slot with the largest number of other packets, or packet  $V_p$  has the highest contention with others. In the contention-based ordering (CBO) algorithm, priority is given to packets with largest  $R_i$  by scheduling it first.

The detailed algorithm is summarized by the following pseudo-codes.

### Contention-based Ordering Algorithm

Input:  $A = [a_{ij}]$   
Output:  $T_1, T_2, \dots, T_L$

1.  $k = 0$ ;
2. Construct compatibility matrix  $C$  from traffic matrix  $A$ ;
3. Find  $R_i$ , the number of 1's in all rows of  $C$ ;
4. If  $A$  is not empty,  $k = k + 1$ ;  $T_k = [0]$ ;  $X = \{\emptyset\}$ ;  
Else  $L = k$  and EXIT;
5. Find  $V_p$ , where  $R_p = \max_i \{R_i, \text{ where } c_{ij} = 0 \forall j \in X\}$ ;
6. If  $R_p > 0$ ,  
add  $V_p$  to  $T_k$ ;  $R_p = "-"$ ;

update  $C$  by setting  $c_{ip} = "-"$  and  $c_{pi} = "-" \forall i$ ;  
 update  $R_i$ ;  
 $X = X + \{p\}$  and goto Step 5;  
 Else goto Step 4;

### Example 1

Consider a  $5 \times 5$  TDM multicast switch with a traffic matrix shown in Fig. 2. From Step 2 of the CBO algorithm, a compatibility matrix is constructed.

$$C = \begin{bmatrix} - & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & - & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & - & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & - & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & - & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & - & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & - & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & - & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & - & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & - \end{bmatrix} \begin{matrix} 4 \\ 7 \\ 5 \\ 4 \\ 4 \\ 7 \\ 4 \\ 6 \\ 7 \\ 4 \end{matrix}$$

$R_i$  the number of 1 entries is shown next to each row of  $C$ . From Step 5, three packets  $V_2, V_6$  and  $V_9$  are found to have the same maximum value of  $R_i = 7$ . In this example, we always choose the last packet in case of a tie. Therefore, packet  $V_9$  is added to the transmission matrix  $T_1$  following Step 6. The compatibility matrix  $C$  is then updated.

$$C = \begin{bmatrix} - & 1 & 0 & 0 & 0 & 1 & 1 & 1 & - & 0 \\ 1 & - & 1 & 0 & 0 & 1 & 1 & 1 & - & 1 \\ 0 & 1 & - & 1 & 1 & 0 & 0 & 0 & - & 1 \\ 0 & 0 & 1 & - & 1 & 1 & 0 & 0 & - & 0 \\ 0 & 0 & 1 & 1 & - & 1 & 0 & 0 & - & 0 \\ 1 & 1 & 0 & 1 & 1 & - & 1 & 1 & - & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & - & 1 & - & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & - & - & 1 \\ - & - & - & - & - & - & - & - & - & - \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & - & - \end{bmatrix} \begin{matrix} 4 \\ 6 \\ 4 \\ 3 \\ 3 \\ 6 \\ 4 \\ 5 \\ - \\ 3 \end{matrix}$$

Then the program loops back to Step 5 for finding a packet that can be transmitted in the same time slot with  $V_9$ . Two packets  $V_1$  and  $V_7$  with  $R_1 = R_7 = 4$  are found. (There are packets with  $R_i \geq 4$  but they have output conflict with  $V_9$ .) Again  $V_7$  is chosen and is added to  $T_1$ .

$$T_1 = \begin{matrix} V_9 \\ V_7 \end{matrix} \left[ \begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

The compatibility matrix is again updated accordingly (not shown). Since no more packet can be added to  $T_1$ , the program loops back to Step 4 and starts to assign  $T_2$  in the same manner. Skipping the details, we have

$$T_2 = \begin{matrix} V_6 \\ V_3 \end{matrix} \left[ \begin{array}{cccc|cccc} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

$$T_3 = \begin{matrix} V_8 \\ V_5 \end{matrix} \left[ \begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right]$$

$$T_4 = \begin{matrix} V_2 \\ V_4 \end{matrix} \left[ \begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right]$$

$$T_5 = \begin{matrix} V_1 \\ V_{10} \end{matrix} \left[ \begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

The frame length is 5 and is the optimal length for this example.

## 4 Hybrid Ordering Algorithm

A column, say  $p$ , in a traffic matrix  $A$  is called a *critical* column if it has the largest number of 1 entries. Then a *critical* traffic vector is a traffic vector which has a value 1 at column  $p$ , or  $a_{ip} = 1$ . In the hybrid ordering algorithm, the critical traffic vector with the largest value of  $R_i$  (i.e. the number of packets with output conflict with the chosen packet) is found first. We call this critical-column based ordering<sup>2</sup> (CCBO). The found critical vector is then assigned to a new/empty transmission matrix. To complete the assignment, other packets are added to this transmission matrix using the contention-based ordering. If no more packet can be added, another new transmission matrix is initiated in a similar fashion. The name hybrid ordering comes from the fact that both critical column-based and contention-based orderings are used in this algorithm.

The above procedures are summarized by the following pseudo-codes.

### Hybrid Ordering Algorithm

Input:  $A = [a_{ij}]$

Output:  $T_1, T_2, \dots, T_L$

1.  $k = 0$ ;
2. Construct compatibility matrix  $C$  from traffic matrix  $A$ ;
3. Find  $R_i$ , the number of 1's in all rows of  $C$ ;
4. Find critical column  $q$  in  $A$ ;  
/\* Critical-column-based ordering \*/
5. Find  $V_p$ , where  $R_p$  is max. for all vectors with  $a_{iq} = 1$ ;  
/\* Contention-based ordering \*/
6. If  $R_p > 0$ ,  $k = k + 1$ ;  $T_k = [0]$ ;  $X = \{\emptyset\}$ ;  
Else  
If  $A$  is empty,  $L = k$  and EXIT;  
Else, goto Step 4;
7. Add  $V_p$  to  $T_k$ ;  $R_p = "-"$ ; /\* 1st packet in  $T_k$  \*/  
update  $C$  by setting  $c_{ip} = "-"$  and  $c_{pi} = "-" \forall i$ ;  
update  $R_i$ ;  
 $X = X + \{p\}$ ;
8. Find  $V_p$ , where  $R_p = \max_i \{R_i, \text{ where } c_{ij} = 0 \forall j \in X\}$ ;
9. If  $R_p > 0$ ,  
add  $V_p$  to  $T_k$ ;  $R_p = "-"$ ;  
update  $C$  by setting  $c_{ip} = "-"$  and  $c_{pi} = "-" \forall i$ ;  
update  $R_i$ ;  
 $X = X + \{p\}$  and goto Step 8;  
Else goto Step 5;

### Example 2

We use the same traffic matrix  $A$  used in Example 1 here. From Step 4, we found column 8 of  $A$  is the critical column since it has 5 1's. The set of critical vectors is  $\{V_1, V_2, V_6, V_7, V_8\}$ . From Step 5,  $V_2$  and  $V_6$  both have the highest contention with other packets. As in Example 1, we choose the last packet in case of a tie. Therefore  $V_6$  is chosen and is added to  $T_1$  in Step 7. The compatibility matrix  $C$  is then updated. Steps 8 and 9 form a loop to find the packets that can be transmitted together with  $V_6$  using the contention-based ordering method. Only  $V_{10}$  is found. So we

<sup>2</sup>In [6], a MTSA algorithm, refer to as critical-column based ordering (CCBO), was proposed.

have

$$T_1 = \begin{matrix} V_6 \\ V_{10} \end{matrix} \left[ \begin{array}{ccccc|ccccc} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array} \right].$$

The program loops back to Step 5 and starts to assign  $T_2$ . Since now  $V_2$  has the largest contention (i.e. the maximum value of  $R_2$ ) with other packets compared to the rest of the critical vectors,  $V_2$  is added to  $T_2$  and the same procedure for assigning  $T_1$  follows. We have

$$T_2 = \begin{matrix} V_2 \\ V_5 \end{matrix} \left[ \begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right]$$

$$T_3 = \begin{matrix} V_8 \\ V_4 \end{matrix} \left[ \begin{array}{ccccc|ccccc} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right]$$

$$T_4 = \begin{matrix} V_7 \\ V_3 \end{matrix} \left[ \begin{array}{ccccc|ccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

$$T_5 = \begin{matrix} V_1 \\ V_9 \end{matrix} \left[ \begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \right].$$

The frame length is 5.

## 5 Performance Evaluation

In this section, we compare the performance of the CBO and hybrid ordering algorithms with three algorithms proposed in the literature. They are the simple greedy (SG) algorithm, divide-and-conquer (DAC) algorithm and critical column-based ordering (CCBO) algorithm [6]. The detailed description of the three algorithms together with illustrated examples can be found in an extended paper [10].

### 5.1 Time Complexity

Simple Greedy algorithm has a time complexity of  $O(L(\alpha^2 N))$ , where  $L$  is the frame length,  $\alpha$  is the number of packets in a traffic matrix, and  $N$  is the switch size. From [6], the time complexities of the DAC and CCBO algorithms are  $O(L(\alpha N + N^{4.5}))$  and  $O(L(\alpha^2 N))$  respectively. The time complexities of the CBO and the hybrid ordering algorithms are both found to be  $O(L(\alpha^2 N))$ .

### 5.2 Simulation Results

Let the number of packets arrived at an input port per traffic matrix be a random integer uniformly distributed between 0 and  $n = 3$ . Let  $\rho$  be the probability that one copy of the generated (multicast) packet will destine for an output port, or fan-out probability. For an  $N \times N$  switch, the average fan-out per input packet is  $N\rho$ . In our simulations, if a packet with no output address is generated, one output port is randomly chosen for it. Each point of the simulation results shown in Figs. 4 to 7 is obtained by averaging over 5000 traffic matrices.

In Figs. 4 and 5, the percentage increase in frame length as compared to the lower bound is shown. The lower bound is the clique number of the MTSA equivalent graph and is found

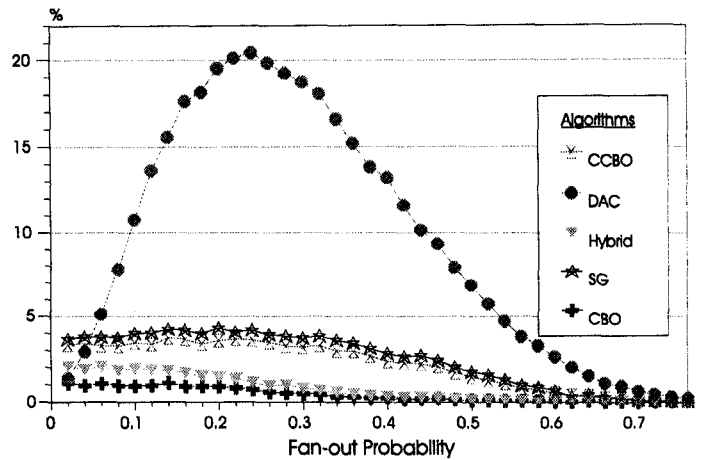


Figure 4: Percentage increase in frame length over the lower bound for a  $5 \times 5$  switch.

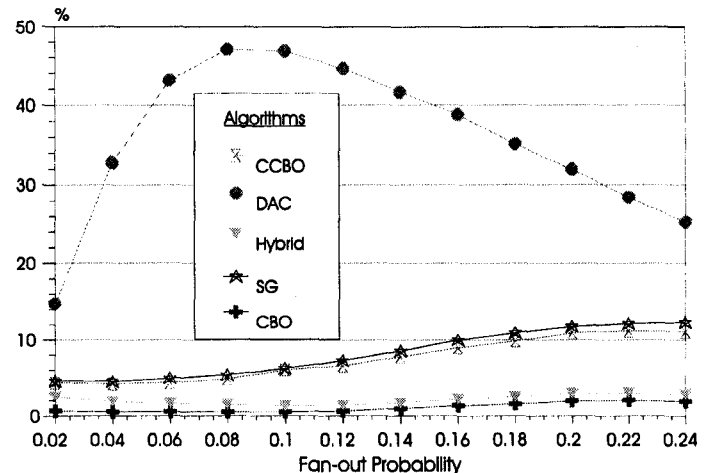


Figure 5: Percentage increase in frame length over the lower bound for a  $15 \times 15$  switch.

by enumerating all possible subgraphs. We can see that the frames generate by both CBO and hybrid ordering algorithms have a very close-to-lower-bound frame length. And this result is quite independent of the fan-out probability. For the CBO algorithm, the percentage increase in frame length over the lower bound is less than 1%. On the other hand, despite the highest complexity, the DAC algorithm gives the poorest performance. The CCBO algorithm is also shown to have only a comparable performance to simple greedy. Compared with the results of DAC and CCBO reported in [6], a large difference in performance is noticed because in [6] switches with only very small sizes were considered. Because of the high complexity in determining clique number, Fig. 6 shows the percentage increase in frame length over the CBO algorithm for a  $25 \times 25$  switch.

For simple greedy, DAC and CCBO, it can be seen that the percentage increase in frame length increases with the switch size. Consider the DAC algorithm as an example, the highest percentage increase in frame length over the lower bound is 21% for  $N = 5$  and 47% for  $N = 15$ . But the performance of the CBO and the hybrid ordering algorithms is shown to be

independent of the switch size. When fan-out probability is high, the performance of all five algorithms converges to the lower bound. This is because no two or more packets can be transmitted in the same time when each packet has a very large fan-out.

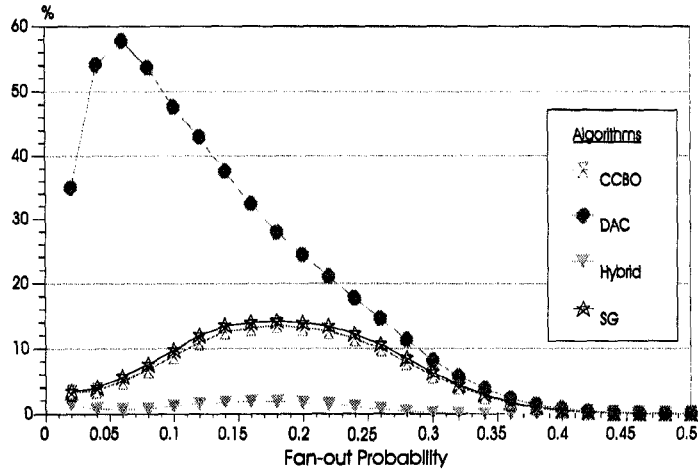


Figure 6: Percentage increase in frame length over that generated by CBO algorithm for a  $25 \times 25$  switch.

In Fig. 7, the performance of the five algorithms under unicast traffic conditions is examined. For unicast time slot assignment problem, many optimal polynomial time algorithms have been proposed. For details please refer to e.g. [11, 2]. The probability that the algorithms failed to generate optimal time slot assignment  $P_F$  is plotted against the switch size. It can be seen that the CBO algorithm again gives the best performance with  $P_F < 0.02$ . For all cases where non-optimal time slot assignments are obtained, we found that only one extra time slot is required.

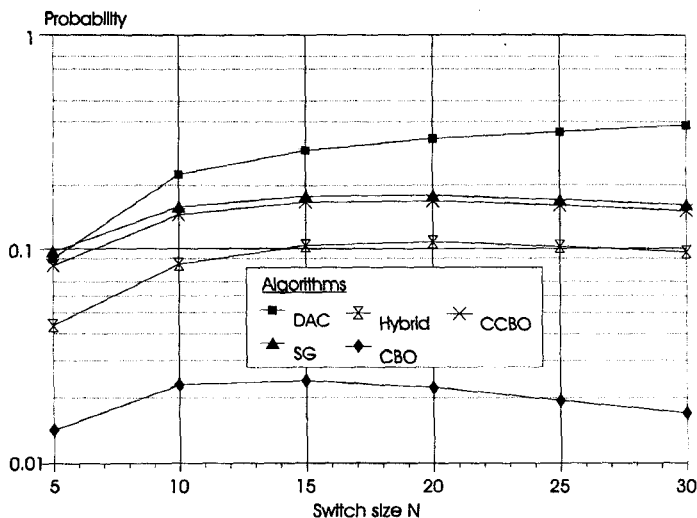


Figure 7: Probability that the algorithms failed to generate optimal time slot assignment under unicast traffic conditions.

## 6 Conclusions

In this paper, we have studied two efficient multicast time slot assignment algorithms, called contention-based ordering (CBO) and hybrid ordering algorithms, for TDM switching systems. A lower bound on the frame length for a multicast time slot assignment was found to be the clique number of a MTSA equivalent graph. The performance of the two proposed algorithms was compared with the three existing algorithms and the lower bound through extensive simulations. We found that the CBO algorithm, which has the lowest complexity, gives the best performance among all five algorithms studied. The average frame length generated by the CBO algorithm is within 1% above the lower bound. We also showed that (i) the previously reported DAC algorithm has the poorest performance despite its highest complexity, and (ii) the previously reported CCBO algorithm has only a comparable performance to the simple greedy algorithm.

The algorithms we proposed in this paper can be extended to multicast packet switching systems. They have potential to give a high system throughput, low mean packet delay and delay variation.

## References

- [1] T. Inukai, "An efficient ss/tdma time slot assignment algorithm," *IEEE Trans. on Commun.*, Vol. COM-27, No. 10, pp. 1449-1455, Oct. 1979.
- [2] C. Rose and M. G. Hluchyj, "The performance of random and optimal scheduling in a time-multiplex switch," *IEEE Trans. on Commun.*, Vol. COM-35, No. 8, pp. 813-817, Aug. 1987.
- [3] S. Chalasani and A. Varma, "An improved time-slot assignment algorithm for tdm hierarchical switching systems," *IEEE Trans. on Commun.*, Vol. 41, No. 2, pp. 312-317, Feb. 1993.
- [4] M. A. Bonucelli, "A fast time slot assignment algorithm for tdm hierarchical switching systems," *IEEE Trans. on Commun.*, Vol. 37, pp. 870-874, Aug. 1989.
- [5] M. Chen and T. S. Yum, "A conflict-free protocol for optical wdma networks," *IEEE Proc. GLOBECOM*, pp. 1276-1281 1991.
- [6] W. Chen, P. Sheu, and J. Yu, "Time slot assignment in tdm multicast switching systems," *IEEE Trans. on Commun.*, Vol. 42, No. 1, pp. 149-165, Jan. 1994.
- [7] F. Box, "A heuristic technique for assigning frequencies to mobile radio nets," *IEEE Trans. Veh. Tech.*, vol. VT-27, pp. 57-64 1978.
- [8] K. N. Sivarajan, R. J. McEliece, and J. W. Ketchum, "Channel assignment in cellular radio," *IEEE Veh. Tech. Conf., VTC'89*, pp. 846-850 1989.
- [9] M. R. Garey and D. S. Johnson, *Computers and Intractability*. W. H. Freeman And Company, 1979.
- [10] K. L. Yeung and K. Au-Yeung, "Efficient time slot assignments for tdm multicast switching systems," *submitted*, 1997.
- [11] K. Y. Eng and A. S. Acampora, "Fundamental conditions governing tdm switching assignments in terrestrial and satellite networks," *IEEE Trans. on Commun.*, Vol. COM-35, No. 7, pp. 755-761, Jul. 1987.