Consiglio Nazionale delle Ricerche

Minimizing the Message Waiting Time in Single-Hop Multichannel Systems

A. Martelli, M. Bonuccelli

IIT TR-24/2010

Technical report

Settembre 2010

Istituto di Informatica e Telematica

Minimizing the Message Waiting Time in Single-Hop Multichannel Systems

Francesca Martelli and Maurizio A. Bonuccelli

***Abstract*—In this paper, we examine the problem of packet scheduling in a single-hop multichannel systems, with the goal of minimizing the average message waiting time. Such an objective function represents the delay incurred by the users before receiving the desired data. We show that the problem of finding a schedule with minimum message waiting time, is NP-complete, by means of polynomial time reduction of the time table design problem to our problem. We present also several heuristics which result in outcomes very close to the optimal ones. We compare these heuristics by means of extensive simulations.**

***Index Terms*—packet scheduling, minimum message waiting time, NP-completeness, heuristics.**

I. Introduction

Single-hop multichannel systems are commonly used in telecommunication networks. Examples of this kind of systems are: satellite-switched time division multiple access, optical networks with passive stars, internet routers and also some wireless networks, such as WiMax and some WiFi LAN (for example, IEEE802.11e) [1], [2], [3], [4], [5], [6], [7].

A system is single-hop when entities share the communication medium and they can communicate directly each other, i.e. without store and forward of messages in intermediate entities. On the converse, in multi-hop systems, entities are distributed on several media and the communication happens through intermediate stations that store and relay the messages.

A single-hop system can be of two types: *singlechannel* or *multichannel*. In a singlechannel system, only one transmission at time can be carried out correctly, like in Ethernet LANs. In a multichannel system instead, the available communication bandwidth is split in several parallel channels (e.g. by dividing the frequency spectrum into subchannels, or by using orthogonal codes) and then multiple stations can communicate simultaneously. So, the systems under investigation in this paper will use FDMA/TDMA or CDMA/TDMA medium access control protocols. In this kind of systems, the packet scheduling problem is of crucial importance to achieve good performances in terms of both bandwidth utilization and delay perceived by the final users.

Packet scheduling problems arise in many different settings, so much work is present in literature. There are papers about the optimization of an objective function, with respect to a set of constraints on the physical switch, such as the number of channels [8], or the bandwidth of the channels [9], or, in case of real-time traffic, the compliance of the deadlines [4]. The typical objective function is the minimization of the schedule length, which is equivalent to maximize the throughput of the systems. Other objective functions are: fairness among users, or minimizing the average packet waiting time.

F. Martelli and M. A. Bonuccelli are with the Department of Computer Science, University of Pisa, Pisa, Italy e-mail: {f.martel,bonucce}@di.unipi.it.

Depending on the type of traffic and the system model, scheduling problems can be divided in *offline* and *online*: in offline models, schedules computation is performed after a packet transmission requests gathering, while in online models, a packet is scheduled as soon as it arrives at the switch, or at the queues before the switch. In online setting, much work has been done about the stability of the system, namely to find conditions and algorithms which avoid the input queues (of finite size) to grow indefinitely [10], [11]. The time a packet remains in the queue before being transmitted, represents the delay that it incurs until it is received by the final user. So the minimization of the packet delay represents a measure of system performance from the point of view of the users [10], [12], [13].

Offline algorithms offer better systems performances, since the schedule is computed on a frame basis. Of course, considering a set of time slots instead of one for user allocations, brings to a better overall utilization of the system. For instance, WiMax systems show how to take advantage of offline scheduling algorithms [14]. Although performance optimization is often possible only by means of slow algorithms, frequently good performances are achieved also with fast suboptimal heuristics.

In this paper, we consider offline algorithms and explore the waiting time problem, focusing on the delay affected by messages, instead of packets. Usually, the final users exchange variable-length messages, which are splitted in equal length packets for being transmitted on the networks. So, from the point of view of the users, it is more important the delay of the last packet of a message, since he/she can not use the message information before receiving it completely (think, for instance, to typical web applications browsing text or images, which are displayed only after being totally received).

This problem has been studied mainly in optical networks, since in such systems the tuning latency is a relevant parameter: in that setting, preemptive schedules are likely longer than non-preemptive ones, because of the tuning latencies for swapping from a wavelength to another one [3], [15].

The problem we face is modeled as follows: time is divided in slots, and a set of consecutive slots forms a frame. For each variable length frame, a traffic matrix represents the requests for data transmission, and on it the scheduling algorithm is applied. A schedule is a set of switching matrices, each one representing the amount of traffic which could be transmitted without conflicts, in one or more consecutive time slots: more precisely, the problem constraints are equivalent to the physical limits of the systems, namely, an input (output) can transmit (receive) only one packet at time, and each channel can carry only one packet at time. The scheduling goal is to minimize the average message waiting time, namely the delay incurred in transmitting the last packet of each message.

In spite of its importance, this problem has received little attention till now, probably for its hardness. In particular, in [12], the minimum packet waiting time problem has been studied in a satellite setting. In that paper, an optimal algorithm which produces minimum length schedules with minimum average packet waiting time has been presented. Such an algorithm is computationally infeasible, since it is based on a branch and bound technique and the running time grows exponentially with the size of the input. In the same paper, some fast heuristics are proposed, which produce solutions very close to the optimal one. For these heuristics, worst case performance bounds are provided, but simulations show that they perform (on the average) much better than the predicted bounds, and produce schedules very close to the optimal one.

In [10], the problem of minimizing the packet delay has been studied from a theoretical point of view, by considering input queued crossbar switches. In such switches, arriving packets are stored in the queues at the inputs before being transmitted. In the paper, the authors show that any scheduling strategy, which does not consider the queue backlog information, produces average delay which is at least $O(N)$ (N being the size of the switch). By contrast, they show that an $O(logN)$ delay is achievable with random inputs, under some constraints on the queue size and the maximum traffic load for the inputs.

The problem of efficiently sequencing variable-length messages has been studied in case of optical networks with passive star [15]. Such a network has a number of channels (wavelengths), and the main scheduling problem is to assign channels to the users. They show that if the channel assignment problem is considered together with the message sequencing problem (namely the transmission order among messages), a better overall system performance can be achieved. About the message sequencing problem, two techniques are taken under consideration for imposing a priority on the order in which messages are transmitted: *longest-job-first*, and *shortest-job-first*. The first technique allows better load balancing among the transmission channels, while the second one allows reduced average delays. In [15] it is shown that the best system performances can be achieved by a proper tradeoff between the two techniques.

The paper is organized in this way: in Section II, we define the model of the system under consideration and formulate the problem to solve; in Section III, we give some properties on the value of the objective function and on the optimal schedules. In Section IV we show that the problem of finding optimal schedules is NP-complete, while in Section V we present some sub-optimal heuristics, that are evaluated by means of simulations in Sections VI and VII . Finally, Section VIII terminates the paper.

II. Problem definition

For the sake of simplicity, we define in this paper the problem of single messages between any pair $(i; j)$, but all results hold for multiple messages, i.e. in the case in which input i has more than one message for output j. Multiple messages between any pair of inputs and outputs has been considered in the simulation experiment. A *traffic matrix* D is an $N \times N$ matrix with nonnegative entries. Let entry $d_{ij} = x > 0$, then we say that input i has a *message* destined to output j which is x packets long. A *line* in a matrix is a column or a row. A *switching matrix* S^k is an $N \times N$ matrix with nonconflicting entries, i.e. no two non-zero entries are on the same line, and it represents a switch configuration for one or more consecutive time slots. Given a traffic matrix D, a *schedule* S is a decomposition $S = \{S^k\}$, $1 \leq k \leq L_S$ of the traffic matrix, such that

$$D = \sum_{k=1}^{L_S} S^k$$

where S^k are switching matrices, and L_S is the *schedule length*. From [16], [8], we recall that the lower bound LB_L on the schedule length L_S is given by

$$LB_L = \max\{r_i, c_j, 1 \leq i, j \leq N\}$$

where

$$r_i = \sum_{j=1}^{N} d_{ij} \text{ and } c_j = \sum_{i=1}^{N} d_{ij},$$

i.e. it is the maximum line sum of the traffic matrix D.

We define $w_S(i, j) = max\{k|S^k(i, j) > 0\}$ as the *waiting time of the message* from input i to output j in schedule S, which is equal to k whenever S^k is the switching matrix in which the last packet of that message is scheduled.

We define as the *total message waiting time* of a schedule S, the following quantity

$$W_S = \sum_{i=1}^{N} \sum_{j=1}^{N} w_S(i, j)$$

Now, we can state the problem subject of this paper:

Minimum Message Waiting Time (MMWT): Given a traffic matrix D, find a schedule S such that the total waiting time W_S is minimum.

In the following we state some properties of the problem, and in Section IV we show that this problem is NP-complete.

III. Properties

In this section, we investigate some properties, such as a lower bound on the W_S value.

We define *modified by rows traffic matrix* D', an $N \times N$ matrix built in the following way: we consider each row at time, and we rearrange the non-zero entries in non-decreasing order. Similarly, we define *modified by columns traffic matrix* D'', an $N \times N$ matrix built by rearranging the non-zero entries of each column in non-decreasing order. In Figure 1, an example of D, D', D'' is shown.

We define *waiting time of row* i *(column* j*)* $wt_r(i)$ ($wt_c(j)$) the progressive sum of the non-zero entries in row i (column j) of matrix D' (D''). Specifically: let nr_i (nc_j) be the number

$$\begin{array}{|c|c|c|}\hline 3 & 2 & \\ \hline & 2 & \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

(a) Traffic matrix, $LB_L = 5$, $LB_W = 15$

$$\begin{array}{|c|c|c|}\hline & & \\ \hline & 1 & \\ \hline \mathbf{1} & & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1 & & \\ \hline & \mathbf{1} & \\ \hline & & \mathbf{1} \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1 & & \\ \hline & & \\ \hline & \mathbf{1} & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline \mathbf{1} & & \\ \hline & & \\ \hline & & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline & 1 & \\ \hline & & \\ \hline & & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline & \mathbf{1} & \\ \hline & & \\ \hline & & \\ \hline\end{array}$$

(b) Optimal schedule S_1, with non minimum length: $W_{S_1} = 1 + 2 + 2 + 3 + 4 + 6 = 18$

$$\begin{array}{|c|c|c|}\hline 1 & & \\ \hline & 1 & \\ \hline & & \mathbf{1} \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1 & & \\ \hline & \mathbf{1} & \\ \hline & & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline \mathbf{1} & & \\ \hline & & \\ \hline & \mathbf{1} & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline & 1 & \\ \hline & & \\ \hline \mathbf{1} & & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline & \mathbf{1} & \\ \hline & & \\ \hline & & \\ \hline\end{array}$$

(c) Minimum length optimal schedule S_2: $W_{S_2} = 1 + 2 + 3 + 3 + 4 + 5 = 18$

$$\begin{array}{|c|c|c|}\hline & 1 & \\ \hline & & \\ \hline \mathbf{1} & & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1 & & \\ \hline & 1 & \\ \hline & & \mathbf{1} \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1 & & \\ \hline & \mathbf{1} & \\ \hline & & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline \mathbf{1} & & \\ \hline & & \\ \hline & \mathbf{1} & \\ \hline\end{array}\quad\begin{array}{|c|c|c|}\hline & \mathbf{1} & \\ \hline & & \\ \hline & & \\ \hline\end{array}$$

(d) Minimum length not optimal schedule S_3: $W_{S_3} = 1 + 2 + 3 + 4 + 5 + 5 + 6 = 25$

Fig. 2. Examples of schedules: a bold entry in position S^k_{ij} represents the last packet of the message between i and j, and gives a contribute of k to the W value.

$$\begin{array}{|c|c|c|}\hline 3 & 2 & \\ \hline & 2 & \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

(a) D

$$\begin{array}{r} nr_1 = 2 \\ nr_2 = 1 \\ nr_3 = 3 \end{array}\quad\begin{array}{|c|c|c|}\hline 2 & 3 & \\ \hline 2 & & \\ \hline 1 & 1 & 1 \\ \hline\end{array}\quad\begin{array}{l} wt_r(1) = 2 + 5 = 7 \\ wt_r(2) = 2 \\ wt_r(3) = 1 + 2 + 3 = 6 \end{array}$$

(b) D'

$$\begin{array}{r} nc_1 = 2 \\ nc_2 = 3 \\ nc_3 = 1 \end{array}\quad\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 3 & 2 & \\ \hline & 2 & \\ \hline\end{array}\quad\begin{array}{l} wt_c(1) = 1 + 4 = 5 \\ wt_c(2) = 1 + 3 + 5 = 9 \\ wt_c(3) = 1 \end{array}$$

(c) D''

Fig. 1. Lower bound computation

of non-zero entries in row i (column j) of matrix D' (D''). Then

$$wt_r(i) = \sum_{p=1}^{nr_i} \sum_{q=1}^{p} d'_{iq}$$

and

$$wt_c(j) = \sum_{p=1}^{nc_j} \sum_{q=1}^{p} d''_{qj}.$$

In particular, the inner sum represents the sum of the waiting times of the messages shorter than the pth one, since it would be scheduled after those. An example of $wt_r(i)$ and $wt_c(j)$ computation is given in Figure 1. Now, we are able to state the following lemma on the lower bound.

Lemma 1: Given a traffic matrix D, a *lower bound* LB_W on the total waiting time for that matrix, when the number of channels available is equal to N, is

$$LB_W = max\left\{\sum_{i=1}^{N} wt_r(i), \sum_{j=1}^{N} wt_c(j)\right\}$$

where $wt_r(i)$ and $wt_c(j)$ are computed on D as described before.

Proof: Clearly, since the contribution to LB_W of a traffic entry (i.e. a message) is given by the position in the schedule of the switching matrix in which the last packet is scheduled, the best way to keep W as low as possible is to schedule smaller entries before larger entries of the traffic matrix. So, consider a row of D: the smallest entry, say d_{ij}, will be scheduled in the first d_{ij} switching matrices; the second smallest, say d_{ik}, will be scheduled after d_{ij}, namely the last packet of d_{ik} will be at least in the $(d_{ij} + d_{ik})$-th switching matrix; and so on. Then, considering the traffic entries by rows (i.e. not considering the constraints on the columns of switching matrices) the contribution of each row i to the total waiting time is given at least by $wt_r(i)$, and for the whole matrix by $\sum_{i=1}^{N} wt_r(i)$. A similar reasoning can be done by considering the columns of the traffic matrix. Then, the lower bound value LB_W for the total waiting time W is given by the maximum value between the two sums. ■

This lower bound on the total waiting time is not tight, namely there are traffic matrices for which it is not achieved. For instance, for the traffic matrix shown in Figure 2, $LB_W = 15$, but a schedule with W smaller than 18 does not exists. In [12], some properties on the schedules for the problem of minimizing the average packet waiting time are given. In particular, it is proved that optimal schedules are always of minimum length. This is not true for the problem of minimizing the average message waiting time, considered in this paper (see Figure 2(b)).

The previous lower bound does not hold when the system has a number of channels (say C) smaller than N. In such a case, no more than C packets can be transmitted simultaneously in the same time slot. In this case, we can compute a lower bound by ignoring the row and column constriants, and considering the availability of C channels in each time slot. Traffic entries are rearranged in increasing order. Then, we build a matrix P of size $C \times 2 * LB_L$ and fill it by considering C traffic entries at time, disregarding the constraints on rows and columns. In Figure 3, we show the computation of this lower bound on traffic matrix of Figure 1(a), when the number of channels available is equal to 2. A bold entry in P_{ij} represents

1	2	3	4	5	6	7	8	9	10
1	**1**	1	**1**						
1	1	**1**	1	1	**1**				

Fig. 3. Lower bound computation in case $C < N$. In the example, $N = 3$, $C = 2$, and D is that one in Figure 1 (a). The lower bound value is given by summing the slot numbers of the last packet of each message: $LB_W = 1 + 1 + 2 + 3 + 4 + 6 = 17$.

a contribution of j units to the lower bound value.

By means of examples, we show in Figure 2 the following two properties:

Property 2: A schedule can be optimal even if it is not of minimum length.
In Figure 2(b), an optimal schedule is shown which is not of minimum length.

Property 3: A minimum length schedule can be not optimal.
In Figure 2(d), a minimum length schedule is shown which is not optimal. As we can see, minimum length schedules can produce very high values of total message waiting time. This makes our problem significantly different from that in [12], in which minimum length schedules are always optimal.

IV. Problem complexity

In this section, we show that the MMWT problem is NP-complete, and in the next section we present some fast heuristics which suboptimally solve the problem in polynomial time. Finally, in Section VI we show that the outputs of the heuristics are close enough to the optimal solution.

Before proving the NP-completeness of the MMWT problem, we notice that the complexity of the *minimum packet waiting time* problem is still open, namely, neither a proof of NP-completeness is given for that problem, nor a polynomial time optimal algorithm is known.

Theorem 4: MMWT problem is NP-complete.

Proof: Clearly, MMWT problem is in NP. To prove the NP-completeness, it is sufficient to find a polynomial time reduction of a known NP-complete problem to it. Consider the following problem [17]:

Timetable Design: Given

1) a finite set $H = \{h_1, ..., h_p\}$,
2) a collection $\{T_1, ..., T_n\}$ where $T_i \subseteq H$, $1 \le i \le n$,
3) a collection $\{C_1, ..., C_m\}$ where $C_j \subseteq H$, $1 \le j \le m$,
4) an $n \times m$ matrix R with nonnegative integer entries r_{ij}.

Question: We ask for a function

$$f(T_i, C_j, h_k) : \{T_1, ..., T_n\} \times \{C_1, ..., C_m\} \times H \to \{0, 1\}$$

such that

1) $f(T_i, C_j, h_k) = 1 \Rightarrow h_k \in T_i \cap C_j$;
2) $\sum_{k=1}^{p} f(T_i, C_j, h_k) = r_{ij}$ for all i and j, $1 \le i \le n$, $1 \le j \le m$;
3) $\sum_{i=1}^{n} f(T_i, C_j, h_k) \le 1$ for all j and k, $1 \le j \le m$, $k \le p$;
4) $\sum_{j=1}^{m} f(T_i, C_j, h_k) \le 1$ for all i and k, $1 \le i \le n$, $k \le p$.

This formulation of the timetable design models the problem of scheduling the teaching program of a school, where H is the set of teaching hours in a week, T_i is the availability of the ith teacher, C_j is the availability of the jth classroom, and r_{ij} is the number of hours the ith teacher must spend in classroom j. The timetable design problem is NP-complete even in the following restricted case [17]:

Restricted Timetable Design (RTT):

1) $p = 3$
2) $C_j = H$, for all j, $1 \le j \le m$
3) $r_{ij} \in \{0, 1\}$, for all i and j, $1 \le i \le n$, $1 \le j \le m$
4) $|T_i| = \sum_{j=1}^{m} r_{ij}$, for all i, $1 \le i \le n$
5) $|T_i| \in \{2, 3\}$.

We transform a generic instance of the restricted timetable design problem into an instance of the MMWT problem in the following way.

$$n = 5, m = 4 \qquad R = \begin{array}{|c|c|c|c|l}
\hline
1 & & 1 & 1 & T_1 = \{1,2,3\} \\ \hline
 & 1 & 1 & & T_2 = \{1,3\} \\ \hline
1 & & & 1 & T_3 = \{1,2\} \\ \hline
 & 1 & & 1 & T_4 = \{2,3\} \\ \hline
1 & 1 & 1 & & T_5 = \{1,2,3\} \\ \hline
\end{array}$$

(a) RTT instance

	1	2	3	4	y_2	z_2	x_4	x_4+1
1	2		2	2				
2		2	2		1	1		
3	2			2				
4		2		2			1	1
5	2	2	2					
a_2					1	1		
b_2					1	1		

(b) Matrix D, after trasformation

Fig. 4. Example of transformation

We build a traffic matrix D, where initially we have one row for each teacher, and one column for each classroom. The matrix is initially filled with zeroes. Then, for each i and j, if $r_{ij} = 1$, we change the value of d_{ij} to 2. Besides, for each teacher i, we add extra lines as follows:

- if $T_i = \{2, 3\}$, then we add two extra columns called column x_i and column $x_i{+}1$, and set $d_{i,x_i} = d_{i,x_i+1} = 1$;
- if $T_i = \{1, 3\}$, then we add two extra columns, say y_i, z_i, and two extra rows, say a_i and b_i. Then, we set d_{a_i,y_i}, d_{b_i,y_i}, d_{a_i,z_i}, d_{b_i,z_i}, and d_{i,y_i} and d_{i,z_i} to 1. All other entries in extra rows and columns are set to 0;
- if $T_i = \{1, 2\}$ or $T_i = \{1, 2, 3\}$, then no line is added.

Let the number of teachers with availability set equal to $\{v, w\}$ be n_{vw}, ($v \in \{1, 2, 3\}$, $w \in \{1, 2, 3\}$ and $v < w$), and those available in all three hours be n_{123}. Obviously, $n_{12} + n_{13} + n_{23} + n_{123} = n$. The final traffic matrix D will then have $n + 2n_{13}$ rows and $m + 2n_{13} + 2n_{23}$ columns.

In Figure 4, an example of the above transformation is given.

We end the transformation by selecting a target value W for the waiting time of matrix D:

$$W = 6n_{12} + 21n_{13} + 13n_{23} + 12n_{123}.$$

Now, we show that the given restricted timetable design problem instance has a solution if and only if the MMWT instance obtained by the above transformation has a schedule whose waiting time is not larger than W. The idea behind the proof is to let the selected teaching hour of teacher i in class j (when $r_{ij} = 1$) correspond to the scheduling of entry d_{ij} ($i \leq n$, $j \leq m$): if teacher i is assigned to class j in the h-th hour, then d_{ij} ($i \leq n$, $j \leq m$) will be scheduled in time slots $2h-1$ and $2h$, and vice versa.

Let us first present some properties of the way entries in D will be scheduled.

If row i of D corresponds to a teacher such that $T_i = \{1, 2\}$, then the only two non-zero entries in such row, say d_{ij} and d_{ik} ($d_{3,1} = d_{3,4}$ in Figure 4 (b)), will be scheduled in this way:

slot number	1	2	3	4
entry scheduled	d_{ij}	$\mathbf{d_{ij}}$	d_{ik}	$\mathbf{d_{ik}}$

and this leads to a contribution to the waiting time of $2+4 = 6$.

If $T_i = \{1, 3\}$ (in Figure 4(b), entries are $d_{2,2} = d_{2,3} = 2$, and $d_{a_2,y_2} = d_{a_2,z_2} = d_{b_2,y_2} = d_{b_2,z_2} = d_{2,y_2} = d_{2,z_2} = 1$), then the scheduling will be:

1	2	3	4	5	6
d_{ij}	$\mathbf{d_{ij}}$	$\mathbf{d_{i,y_i}}$	$\mathbf{d_{i,z_i}}$	d_{ik}	$\mathbf{d_{ik}}$
$\mathbf{d_{a_i,y_i}}$	$\mathbf{d_{b_i,y_i}}$				
$\mathbf{d_{b_i,z_i}}$	$\mathbf{d_{a_i,z_i}}$				

Fig. 5. Schedule for entries related to $T_i = \{1, 3\}$

which contributes $2+3+4+6+1+1+2+2 = 21$ to the waiting time.

If $T_i = \{2, 3\}$ ($d_{4,2} = d_{4,4} = 2$, and $d_{4,x_4} = d_{4,x_4+1} = 1$ in Figure 4(b)), the scheduling will be

1	2	3	4	5	6
$\mathbf{d_{i,x_i}}$	$\mathbf{d_{i,x_i+1}}$	d_{ij}	$\mathbf{d_{ij}}$	d_{ik}	$\mathbf{d_{ik}}$

with a contribution of $1 + 2 + 4 + 6 = 13$ to the waiting time.

Finally, when $T_i = \{1, 2, 3\}$ (in our example of Figure 4 are the entries related to T_1 and T_5), we will schedule the entries in this way:

1	2	3	4	5	6
d_{ij}	$\mathbf{d_{ij}}$	d_{ik}	$\mathbf{d_{ik}}$	d_{il}	$\mathbf{d_{il}}$

with a contribution of $2 + 4 + 6 = 12$ to the waiting time. Notice that the entries with value 2 (those in the non-extra lines) can be swapped without altering the schedule.

Let us assume now that the given instance of RTT has a solution. Then, if teacher i is assigned to class j during hour h, we schedule the corresponding entry d_{ij} in time slots $2h-1$ and $2h$ whenever the case, we schedule the entries in the extra rows or columns according to the above schemata. For instance, if $T_i = \{1, 3\}$, then we schedule d_{ij} (d_{ik}) in time slots 1 and 2, if $f(T_i, C_j, 1) = 1$ ($f(T_i, C_k, 1) = 1$), and we schedule it in slots 5 and 6 if $f(T_i, C_j, 3) = 1$ ($f(T_i, C_k, 3) = 1$). In order for the above scheduling to be legal, in each time slot we must have at most one entry from the same line.

This is true for the extra lines, by construction, since, if $T_i = \{2, 3\}$ we have only one entry per extra column, and if $T_i = \{1, 3\}$, the scheduling shown in Figure 5 meets the above constraint, because d_{i,y_i}, d_{i,z_i}, d_{a_i,y_i}, d_{b_i,y_i}, d_{a_i,z_i}, d_{b_i,z_i} are the only non-zero entries in such extra lines.

The same holds for the entries not in extra lines, also. In fact, entries in the same row are scheduled in different time slots (see the above figures). If two entries in the same column, say d_{ij} and d_{lj}, are scheduled in the same slots, then both teachers i and l would have been assigned to class j during the same hour, and so RTT would have not been solved, a contradiction. The total waiting time of the above schedule is $W = 6n_{12} + 21n_{13} + 13n_{23} + 12n_{123}$, as can be easily checked.

Let us assume now that the scheduling problem obtained from the above transformation applied to the given RTT problem instance, has a solution with a waiting time not larger than W. Then, the only way of obtaining a schedule of waiting time not larger than W is by scheduling the entries according to the above schemata: this is obvious for all the cases but for $T_i = \{1, 3\}$. For such a case, the alternative schedules would schedule the entries equal to 1 in row i earlier, or later. In the case they are both scheduled earlier, or if one is scheduled earlier and the other later, then the waiting time would be at least 22 instead of 21 (see Figures 6 and 7).

1	2	3	4	5	6
$\mathbf{d_{i,y_i}}$	$\mathbf{d_{i,z_i}}$	d_{ij}	$\mathbf{d_{ij}}$	d_{ik}	$\mathbf{d_{ik}}$
$\mathbf{d_{a_i,z_i}}$	$\mathbf{d_{b_i,y_i}}$	$\mathbf{d_{a_i,y_i}}$			
		$\mathbf{d_{b_i,z_i}}$			

Fig. 6. Both earlier: waiting time contribution= 22.

1	2	3	4	5	6
$\mathbf{d_{i,y_i}}$	d_{ij}	$\mathbf{d_{ij}}$	$\mathbf{d_{i,z_i}}$	d_{ik}	$\mathbf{d_{ik}}$
$\mathbf{d_{a_i,z_i}}$	$\mathbf{d_{a_i,y_i}}$	$\mathbf{d_{b_i,y_i}}$			
	$\mathbf{d_{b_i,z_i}}$				

Fig. 7. One earlier and one later: waiting time contribution= 22.

It is easy to see that if we let d_{ij} correspond to r_{ij}, and if we assign teacher i to class j in hour h when d_{ij} is scheduled in slots $2h-1$ and $2h$, the RTT instance has a solution. In fact, all the four constraints of RTT are met: no teacher is assigned when not available, all requirements r_{ij} are met, and at most one teacher is assigned to a class in each hour. ■

The above NP-completeness result practically leaves us with the choice between a slow (exponential time) optimal algorithm, or fast but suboptimal heuristics. In the next section, we present some simple heuristics which bring to sub-optimal solutions.

V. Heuristics

In this section, we describe three simple heuristics which solve the MMWT problem in polynomial time. Two of them are of "greedy" type, while the third one is based on maximum cardinality minimum weight matching algorithm.

A. Greedy (GRE)

This is a very simple heuristic. The schedule is built in this way: non-zero entries in the traffic matrix are considered in

5	3	1
1	1	4
2	2	1

(a) Traffic matrix D

9	6	2
2	2	6
5	5	2

(b) Q matrix associated with D

Fig. 8. Example of Q matrix

increasing order. A switching matrix is composed by choosing the minimum values first that are in lines both currently exposed (namely, with only zero entries) in such matrix. After building it, the switching matrix is subtracted from the traffic matrix, and the procedure is repeated until the traffic matrix is empty (only zero entries). Pseudocode of this heuristic follows.

Greedy Algorithm:

```
Step 1: initialization
$k \leftarrow 1$;
Step 2: main loop
while($D$ not empty)
  Step 3: variables setup
  $D' \leftarrow D$;
  Step 4: build the $k$th switching matrix
  while($D'$ not empty)
      Step 5: entry selection for the $k$th switching matrix
      find $i, j$ such that $D'_{ij}$ is minimum;
      $S^k_{ij} = D_{ij}$;
      Step 6: clear row $i$ and column $j$ of $D'$
      for($p = 1$ to $N$) $D'_{pj} = 0$;
      for($p = 1$ to $N$) $D'_{ip} = 0$;
  end while
  Step 7: variables update
  cut non-zero entries in $S^k$ to the minimum value $x$;
  $D \leftarrow D - S^k$;
  $k \leftarrow k + x$;
end while
```

About the time complexity of this algorithm, the main loop runs at most r times, where r is the number of non zero entries in the traffic matrix D, since at each time at least one of them becomes zero. Given that each computation in the main loop is $O(N^2)$, and r is at most N^2, the total time complexity of GRE heuristic is $O(N^4)$.

B. Dynamic greedy (DG)

This heuristic is similar to the previous greedy algorithm, with the following difference: instead of considering the non-zero entries for their value, we build a matrix Q which represents the lower bound on the contribution of each entry to the total waiting time. Specifically, each non-zero entry d_{ij} is replaced with a value q_{ij} which is obtained by chosing the maximum value between the sum of d_{ij} with lower or equal values in its row, and the sum of d_{ij} with lower or equal values in its column.

An example of Q matrix is shown in Figure 8. Matrix Q is then re-computed after each switching matrix generation on the residual traffic matrix. *Step 3* and *Step 5* of the previous greedy algorithm are modified in the following way:

Dynamic Greedy Algorithm:

```
  Step 3: variables setup
  $D' \leftarrow D$;
  compute $Q(D')$;

      Step 5: entry selection for the $k$th switching matrix
      find $i, j$ such that $Q_{ij}$ is minimum;
      $S^k_{ij} = D_{ij}$;
```

The time complexity of this heuristic is $O(N^4 \log N)$, since the computing of Q requires $O(N^2 \log N)$ time.

C. Max-Min Matching (MMM)

This heuristic follows a different approach with respect to the previous ones. Instead of greedy selection of entries, switching matrices are computed by applying the maximum cardinality minimum weight matching algorithm [18]. For keeping the total waiting time as much low as possible, it is needed to schedule small entries in the traffic matrix before larger entries. This heuristic aims to build the first switching matrices with the largest number of small entries. To do that, it recursively applies the max-min matching algorithm to the traffic matrix until it is empty.

Max-min Algorithm:

```
Step 1: initialization
$k \leftarrow 1$;
Step 2: main loop
while($D$ not empty)
  Step 3: build the $k$th switching matrix
  find a max-min matching $M$ on $D$;
  Step 4: $S^k$ computation
  $S^k_{ij} = M_{ij}$;
  Step 5: variables update
  cut non-zero entries in $S^k$ to the minimum value $x$;
  $D \leftarrow D - S^k$;
  $k \leftarrow k + x$;
end while
```

In the best of our knowledge, max-min matching can be computed in $O(N^{2.5})$ [19], and it is performed at most $O(N^2)$ times. So, the time complexity of this heuristic is $O(N^{4.5})$, namely greater than the previous greedy algorithms, but it achieves better performance, as we shall see in the next section.

VI. Simulations

In this section, we show the behaviour of the above heuristics compared among them, with respect to an exponential time optimal algorithm, and also with other known heuristics.

We implemented the algorithms in C language, compiled with *gcc* on a linux machine with Fedora as operating system. For each heuristic, we also implemented a *non-preemptive* version, to evaluate their behavior when used in those systems in which preemption has a high cost in terms of time [2], [3]. For that case, we shall call them GRE_{NP}, DG_{NP},

MMM_{NP}, respectively. Non-preemption is achieved in the following way: when the first packet of a message is assigned to switching matrix S^k, also the subsequent $m-1$ packets of that message are assigned to switching matrices S^{k+i}, $1 \leq i \leq m-1$. Consequently, for the greedy heuristics GRE_{NP} and DG_{NP}, *Step 3* is modified such that the k^{th} D' matrices have zero lines whenever S^k has already covered lines. In *Step 5*, when a message D_{ij} has been selected, a packet is placed in each one of the D_{ij} subsequent switching matrices. And in *Step 7*, each scheduled message is removed from D_{ij}. Below, the details in pseudo-code.

```
Step 3: variables setup
$D' \leftarrow D$;
for $(i = 1; i < N; i++)$
    for $(j = 1; j < N; j++)$
        if $(S^k_{ij}! = 0)$ then $D'_{ij} = 0$;


    Step 5: entry selection for the $k$th switching matrix
    find $i, j$ such that $D'_{ij}$ (or $Q_{ij}$) is minimum;
    for($p = 0$; $p < D'_{ij}$; $p++$)
        $S^{k+p}_{ij} = 1$;


Step 7: variables update
$D \leftarrow D-$ messages scheduled in Step 5;
$k \leftarrow k + 1$;
```

Similarly, for the heuristic based on max-min matching.

For comparison purposes, besides an optimal algorithm, we implemented also:

- the BCW algorithm [8], which always produces minimum length schedules;
- an heuristic which builds the schedule in a totally random way.

In the following, we give some details about these algorithms, and the optimal one.

A. Optimal algorithm (OPT)

For the sake of completeness, we describe here an exponential-time optimal algorithm to find an MMWT schedule. It is based on a branch-and-bound procedure, where each node of the tree represents the residual traffic matrix after the generation of a switching matrix. The root is the initial traffic matrix, and each node has a number of sons equal to the number of possible switching matrices, namely $N!$. The algorithm starts with a total waiting time value W_{S_0} computed offline by an heuristic (for instance, the previous greedy algorithm). When a switching matrix is generated, we compute the waiting time of the messages scheduled so far, and the lower bound on the residual traffic matrix: if the sum of these values is greater than W_{S_0}, then that node becomes a leaf, and that branch is pruned since of course the schedules obtained from that branch will not be optimal. Otherwise, the computation is continued on that branch with the new value of W_{S_0}.

Due to the exponential nature of this algorithm, it has been evaluated only in those simulation tests for which the switch size N is small.

B. BCW algorithm (BCW)

This algorithm [8] always produces minimum length schedules. It has been implemented to show that, in this problem, long computations for producing minimum length schedules result in performances which are worse than the fast greedy heuristics. This algorithm is based on the Birkoff-Von Neumann theorem which asserts that a quasi-double stochastic matrix (namely, one in which the line sums are all equal to the same value) is decomposable in permutation matrices, which represent the switching matrices of a schedule. The algorithm adds some dummy traffic to the traffix matrix for making it a quasi-doubly stochastic one. This dummy traffic is then removed from the output. Time complexity is $O(N^{4.5})$. For more details on this algorithm, see [8].

C. Random algorithm ($RAND$)

This heuristic is the simplest algorithm that could be implemented to solve our problem. Regardless of its size, a non zero entry in the traffic matrix is randomly chosen and placed in the schedule, by meeting only the constraints on the switching matrix lines.

Time complexity of this heuristic is equal to that one of greedy algorithm, namely $O(N^4)$. This algorithm is compared with the others to see if it is worthwhile the effort of using some intelligence, or not.

VII. Simulation results

We performed extensive simulations, by tuning all possible problem parameters. In particular, we tuned the following parameters:

- the size of the switch N;
- the number of available transmission channels C, $C \leq N$;
- the maximum number of messages K between any pair of input/output;
- the maximum length of the messages, M;
- the traffic matrix $sparsity$, namely the percentage of zero entries.

We show the results by means of *efficiency* E as performance metric, which is defined as the ratio between the total message waiting time W_S and the lower bound LB_W given in Section III.

Traffic entries in D matrices have been generated always following the uniform distribution, according to the $sparsity$ set for each test. Each test ran 100 times and we show in the graphs the average efficiency values. We computed also the 95% confidence intervals: due to space limitations, we show the values for one test case in Table I.

A. Message waiting time vs. switch size

In Figures 9 and 10 we show the average behavior of heuristics by varying the switch size N, when traffic matrix sparsity is equal to 75% and 0%, respectively.

Algorithm	95% confidence interval
GRE	1368561.43 ± 2572.57
DG	1364624.27 ± 2562.03
MMM	1336687.17 ± 2538.58
GRE_{NP}	1553142.79 ± 3130.06
DG_{NP}	1554936.59 ± 3048.75
MMM_{NP}	1345057.36 ± 2546.63
BCW	2148887.78 ± 5131.77
$RAND$	2146724.61 ± 4189.08

TABLE I
AVERAGE TOTAL MESSAGE WAITING TIMES WITH 95% CONFIDENCE INTERVALS. $N = 128$, $C = 128$, $K = 1$, $M = 5$, $sparsity = 25\%$.

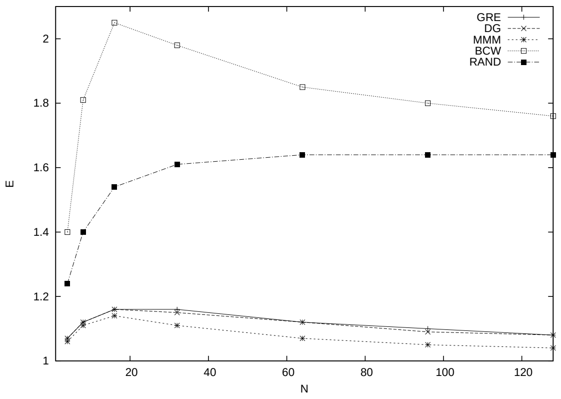

Fig. 9. Efficiency vs N, $C = N$, $K = 1$, $M = 5$, $sparsity = 75\%$.

As we can see, with very sparse traffic matrices, the algorithms efficiencies are worse than with very dense traffic matrices. We notice also that the efficiency increases with the increasing of the switch size and of the decreasing of the traffic matrix sparsity.

B. Message waiting time vs. channel availability

In Figures 11 and 12, we show the behaviour of the efficiency of the algorithms by varying the number of channels available for transmission. We have considered a switch of size $N = 32$ and we tuned the number of channels C from 2 to 32. Notice that our MMWT problem is not much sensitive to this parameter.

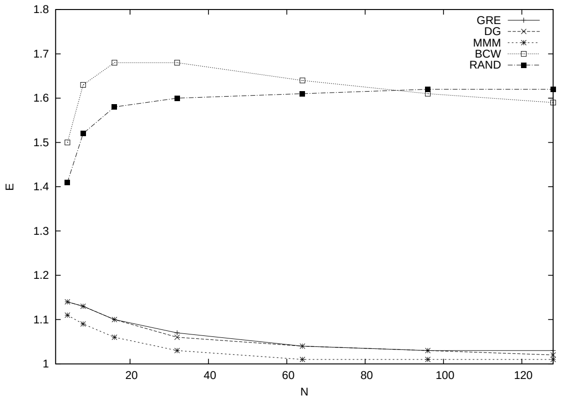

Fig. 10. Efficiency vs N, $C = N$, $K = 1$, $M = 5$, $sparsity = 0\%$.

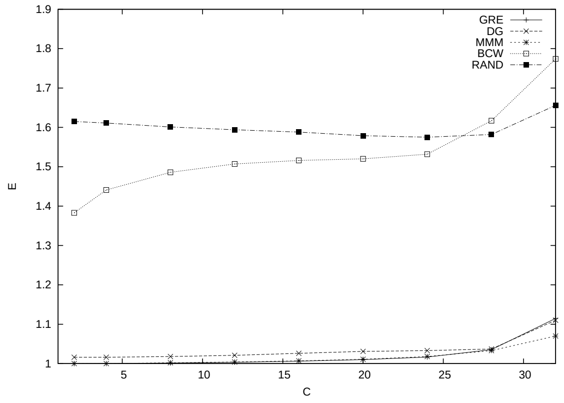

Fig. 11. Efficiency vs number of channels available C, $N = 32$, $K = 1$, $M = 5$, $sparsity = 25\%$.

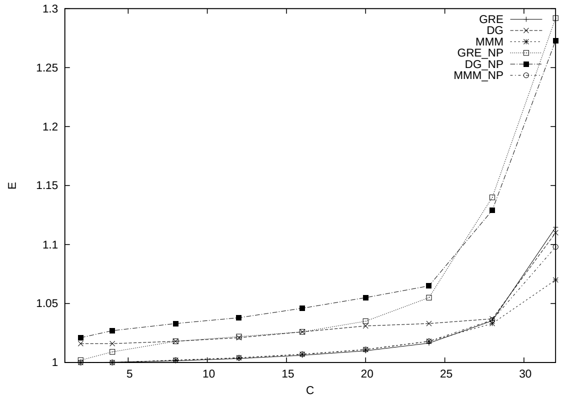

Fig. 12. Efficiency vs number of channels available C, $N = 32$, $K = 1$, $M = 5$, $sparsity = 25\%$.

C. Message waiting time vs. number of messages

In Figure 13, we plotted only the three proposed heuristics and their respective non preemptive versions with respect to the number of messages between any pair of input/output. We notice that the MMM_{NP} heuristic is not much sensitive to this parameter, and its performance is a little lower than the MMM heuristic. For the greedy heuristics instead, non preemptive versions perform worse than preemptive algorithms, with a loss of about 15% in terms of efficiency.

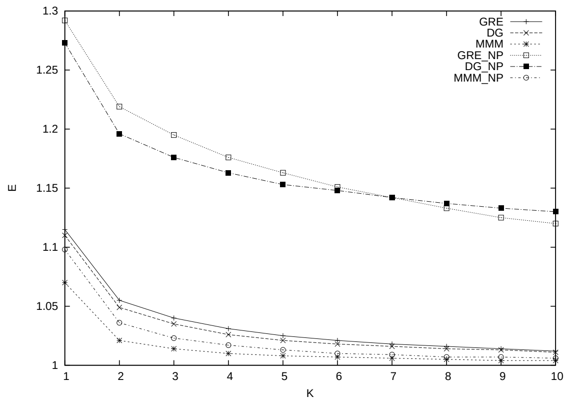

Fig. 13. Efficiency vs number of messages K, $N = C = 32$, $M = 5$, $sparsity = 25\%$.

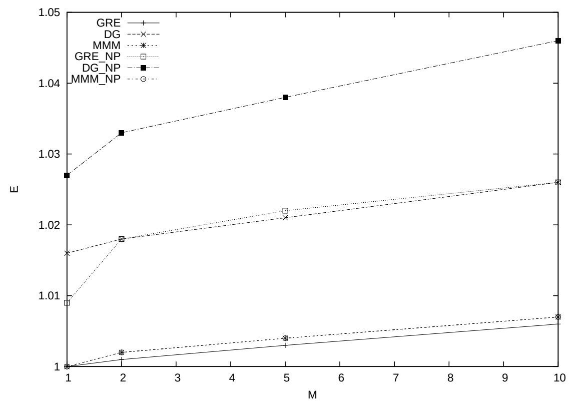

Fig. 14. Efficiency vs message size M, $N = C = 32$, $K = 3$, $sparsity = 25\%$.

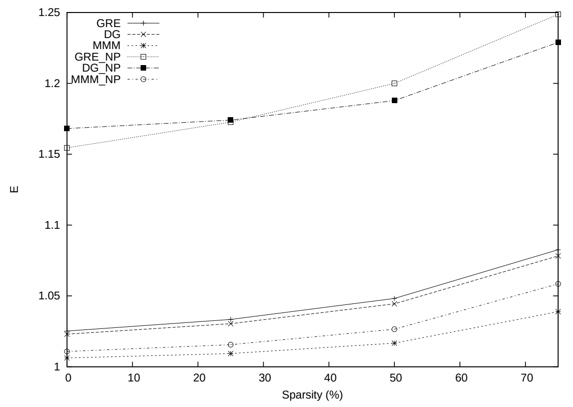

Fig. 15. Efficiency vs traffic matrix $sparsity$, $N = C = 128$, $K = 1$, $M = 5$.

D. Message waiting time vs. length of messages

In Figure 14, we show the efficiency values obtained by changing the message size parameter. We note that the MMWT problem is not sensitive to this parameter.

E. Message waiting time vs. traffic matrix sparsity

Figure 15 shows the behaviour of the efficiency for four values of traffic matrix sparsity: 0%, 25%, 50% and 75%. Notice that for all the heuristics, efficiency decreases with very sparse traffic matrices.

F. Optimal algorithm

In Figure 16, we show the behavior of the heuristics together with the optimal algorithm, for small values of N. As we can see, the efficiency of the MMM heuristic is very close to that one of the optimal algorithm.

G. Other statistical data

In this section, we present other metrics that have been evaluated in the simulations. We have computed, for each heuristic, the following amounts:

- the number of schedules with optimal W_S value (W_{opt});
- the maximum gap between the lower bound LB_W and the obtained total message waiting time W_S (MaxGapW);

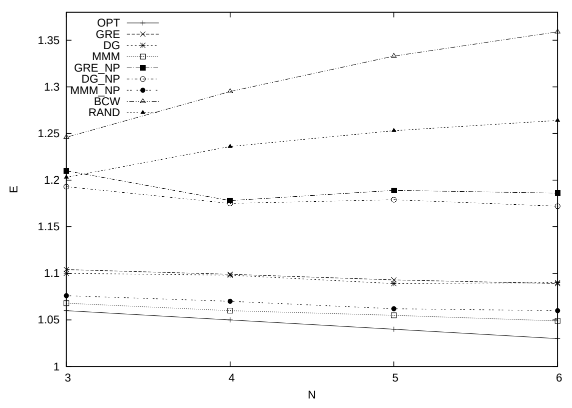

Fig. 16. Efficiency vs N (small values), $C = N$, $K = 1$, $M = 2$, $sparsity = 0\%$.

- the average gap between LB_W and W_S, expressed in percentage (AveGapW);
- the number of schedules of minimum length (L_{opt});
- the maximum gap between the minimum length schedule and the length of obtained schedules (MaxGapL);
- the average gap between LB_L and the length of obtained schedules, expressed in percentage (AveGapL).

For the sake of conciseness, we summarize these results in Tables II and III. Notice that MMM heuristic improves its performance with high values of N, leading to a less than 1% of performance degradation with respect to the lower bound, both in terms of total waiting time values and of schedule lengths. Remember that the AveGapW is computed on the lower bound, and not on the optimal value. A similar consideration holds for MMM_{NP} heuristic also, which results to provide good schedules even with the non preemption constraint. So, we conclude that these heuristics give very good sub-optimal solutions to the MMWT problem.

VIII. Conclusions

In this paper we studied the problem of minimizing the average message waiting time in a single-hop multichannel system. We have shown that this problem is NP-complete, and we proposed and analyzed three fast heuristics. By means of simulations, we have obtained the performances of the proposed heuristics, compared with other algorithms. We realized that the MMWT problem can be solved with algorithms which produce schedules very close to the optimal one. In particular, the MMM heuristic is the most suitable for optimizing the efficiency both for the system and for the users: for the system, because it produces schedules very often of minimum length, and for the users, since the average message waiting time is very close to the optimal one.

Some problems are still open: for instance, the complexity of minimizing the average packet waiting time (studied in [12]) is unknown. A stimulating research is the MMWT problem in a real-time setting, namely when messages have deadlines to be met.

Algorithm	W_{opt}	MaxGapW	AveGapW (%)	L_{opt}	MaxGapL	AveGapL (%)
OPT	100	7	3.54	100	0	0.0
GRE	3	18	9.36	43	2	7.77
DG	1	17	8.95	42	3	9.05
MMM	27	12	5.53	69	2	3.86
BCW	0	64	33.43	100	0	0.0
$RAND$	0	38	25.27	65	2	4.21
GRE_{NP}	0	39	18.88	1	5	28.3
DG_{NP}	1	32	17.77	2	6	31.33
MMM_{NP}	21	13	6.17	65	2	3.99

TABLE II
COMPARISON FOR OTHER METRICS, WITH $N = 5$, $C = 5$, $K = 1$, $M = 2$ AND $sparsity = 0\%$.

Algorithm	W_{opt}	MaxGapW	AveGapW (%)	L_{opt}	MaxGapL	AveGapL (%)
GRE	0	61166	2.52	2	21	1.8
DG	0	56120	2.3	0	22	2.59
MMM	0	15242	0.62	38	7	0.31
BCW	0	1513630	58.91	100	0	0.0
$RAND$	0	1472956	61.66	53	4	0.15
GRE_{NP}	0	383038	15.45	0	122	21.87
DG_{NP}	0	409422	16.81	0	153	27.87
MMM_{NP}	0	27014	1.08	31	7	0.39

TABLE III
COMPARISON FOR OTHER METRICS, WITH $N = 128$, $C = 128$, $K = 1$, $M = 5$ AND $sparsity = 0\%$.

REFERENCES

[1] M. Bonuccelli, I. Gopal, and C. Wong, “Incremental time slot assignment in ss/tdma satellite systems,” *IEEE Transactions on Communication*, vol. 39, no. 7, pp. 1147–1156, July 1991.

[2] H. Choi, H. Choi, and M. Azizoglu, “Efficient scheduling of transmissions in optical broadcast networks,” *IEEE/ACM Transactions on Networking*, vol. 4, no. 6, pp. 913–920, December 1996.

[3] R. Cruz and S. Al-Harthi, “A service-curve framework for packet scheduling with switch configuration delays,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 196–205, February 2008.

[4] Y. Lee, J. Lou, J. Luo, and X. Shen, “An efficient packet scheduling algorithm with deadline guarantees for input-queued switches,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 212–225, February 2007.

[5] H-Y.Wei, S. Ganguly, R. Izmailov, and Z. Haas, “Interference-aware ieee 802.16 wimax mesh networks,” vol. 5, 2005, pp. 3102–3106.

[6] A. Zaki and A. Fapojuwo, “Efficient scheduling algorithms for multi-service multi-slot ofdma networks,” vol. 1, April 2009, pp. 1–6.

[7] Q. Zhao and D. H. K. Tsang, “An equal-spacing-based design for qos guarantee in ieee 802.11e hcca wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 7, no. 12, pp. 1474–1490, December 2008.

[8] G. Bongiovanni, D. Coppersmith, and C. K. Wong, “An optimal time slot assignment algorithm for a SS/TDMA system with variable number of transponders,” *IEEE Transactions on Communications*, vol. 29, no. 5, pp. 721–726, May 1981.

[9] P. Barcaccia and M. Bonuccelli, “A polynomial time optimal algorithm for time slot assignment in variable bandwidth systems,” *ACM/IEEE Transactions on Networking*, vol. 2, no. 3, pp. 247–251, March 1994.

[10] M. J. Neely, E. Modiano, and Y.-S. Cheng, “Logarithmic delay for $n \times n$ packet switches under crossbar constraint,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 657–668, June 2007.

[11] P. R. Kumar and S. P. Meyn, “Stability of queueing networks and scheduling policies,” *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 251–260, February 1995.

[12] I. Gopal, D. Copperswmith, and C. K. Wong, “Minimizing packet waiting time in a multibeam satellite system,” *IEEE Transactions on Communications*, vol. COM-30, no. 2, pp. 305–316, February 1982.

[13] J. Bruno, E. C. Jr., and R. Sethi, “Scheduling independent tasks to reduce mean finishing time,” *Communications of the ACM*, vol. 17, no. 7, pp. 382–387, July 1974.

[14] I. S. 802.16-2004, *IEEE standard for local and metropolitan area networks part 16: air interface for fixed broadband wireless access systems*, 2004.

[15] B. Hamidzadeh, M. Maode, and M. Hamdi, “Efficient sequencing techniques for variable-length messages in WDM networks,” *Journal of Lightwave Technology*, vol. 17, no. 8, pp. 1309–1319, August 1999.

[16] T. Inukai, “An efficient SS/TDMA time slot assignment algorithm,” *IEEE Transactions on Communications*, vol. 27, no. 10, pp. 1449–1455, October 1979.

[17] S. Even, A. Itai, and A. Shamir, “On the complexity of timetable and multicommodity flow problems,” *SIAM Comput.*, vol. 5, pp. 691–703, December 1976.

[18] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[19] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*. Prentice Hall, 1993.