Minimum Message Waiting Time Scheduling in Distributed Systems

Francesca Martelli and Maurizio A. Bonuccelli

Abstract—In this paper, we examine the problem of packet scheduling in a single-hop multichannel system, 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.

1 Introduction

The problem of allocating limited communication resources among competing entities, is becoming more and more relevant with the increasing utilization of parallel and distributed systems, both for communication and for computing [1]. The communication portion of many parallel and distributed systems, as well as communication networks, or part of them, is formed by single-hop multichannel systems. Examples of this kind of systems are interconnection networks, like Benes or Clos networks (used for instance in the best performing internet routers), optical networks with passive stars, and also some wireless networks, such as WiMax and some WiFi LAN (for example, IEEE802.11e) [15], [3], [4], [5], [6], [7], [10].

A system is single-hop when entities share the communication medium, and 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 communication medium, such as the number of channels [11], or precedence constraints on messages [12], or, in case of real-time traffic, the compliance of the deadlines [5]. Another important feature, mainly studied in optical networks, is the tuning latency: in that setting, preemptive schedules are likely longer than non-preemptive ones, because of the time needed for swapping from a wavelength to another one [4], [19]. 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.

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 communication subsystem, or at the queues before such subsystem. 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 [18]. 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 [18], [17], [16].

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 just one (as in online algorithms), 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 [25]. Although performance

F. Martelli is with IIT-CNR, Pisa, Italy e-mail:francesca.martelli@iit.cnr.it
M. A. Bonuccelli is with the Department of Computer Science, University of Pisa, Pisa, Italy e-mail:bonucce@di.unipi.it

optimization is often possible only by means of slow algorithms, frequently good performances are achieved also with fast sub-optimal 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 often consist of computer files, 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). The delay incurred in the communication subsystem is only part of the total delay, but it is a considerable one.

The problem we face is modeled as follows: time is divided in slots, and a set of consecutive slots forms a frame. Consecutive frames can be formed by a different number of time slots. 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. Observe that the complimentary problem of minimizing the maximum message waiting time is equivalent to the classical and well studied problem of minimizing the schedule length (in fact, the schedule length is given by the time the last message is scheduled to completion).

The paper is organized in this way: in Section 2, we define the model of the system under consideration, and formulate the problem to solve; in Section 3 we present the relevant work published on our and on strictly related problems, while in Section 4, we give some properties on the value of the objective function and on the optimal schedules. In Section 5 we show that the problem of finding optimal schedules is NP-complete, while in Section 6 we present some sub-optimal heuristics, that are evaluated by means of simulations in Sections 7 and 8 . Finally, Section 9 terminates the paper.

2 Problem definition

For the sake of simplicity, we define in this section the minimum message waiting time problem for single messages between any communicating pair $(i; j)$. All the results easily extend to the multiple messages case, i.e. the case in which input i has more than one message for output j. Multiple messages between pairs of inputs and outputs has been considered in the simulation experiments. Similarly, we explicitly consider symmetric systems, where the number of inputs is equal to the number of outputs, denoted with N. Again, it is easy to see that all results extend to the case of asymmetrical systems, where we have N inputs and M outputs. 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. In general, the communication subsystem can have a limited number C of channels, $C < N$, thus allowing the simultaneous transmission of at most C packets. A *switching matrix* S^k is an $N \times N$ matrix with entries equal to 0 or 1, such that no two non-zero entries are on the same line, and there are at most C non-zero entries. It represents a switch configuration for one time slot. Given a traffic matrix D, a *legal schedule* (or simply *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 [11], [20], we recall that the lower bound LB_L on the schedule length L_S is given by

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

where

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

i.e. it is the maximum between line sums (r_i for row i, and c_j for column j), and the total traffic divided by the number of internal channels, 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: it is equal to k whenever S^k is the switching matrix in which the last packet of that message is scheduled.

The *total message waiting time* of a schedule S is given by

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

Now, we can state the problem studied in 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 5 we show that this problem is NP-complete.

3 Related work

In spite of its importance, this problem has received little attention till now, probably for its hardness. In particular, in [17], the minimum packet waiting time problem has been studied in a setting of satellites equipped with an interconnection network. 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 very slow, 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 [18], the problem of minimizing the packet delay has been studied from a theoretical point of view, by considering input queued crossbar switches, and with the objective of investigating the queue length due to different scheduling policies. 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 analysis of delay in priority queues has been investigated in [8]. In that paper, the delay of individual packets is addressed. Such packets can belong to two different classes, one having higher priority over the other. New scheduling mechanisms for controlling, in an exact way, the delay differentiation between the two classes, are proposed and analyzed. In a related paper [9], the same assumptions on traffic (namely, packets belonging to two different priority classes) have been made, and analytic techniques to compute delay characteristics of the two classes, are presented.

The problem of efficiently sequencing variable-length messages has been studied in case of optical networks with passive star [19]. 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 [19] it is shown that the best system performances can be achieved by a proper tradeoff between the two techniques.

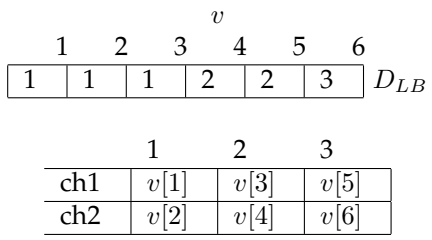


Fig. 1. Lower bound computation in the general case $C \leq N$. In the example, $N = 3$, $C = 2$, and D is that one in Figure 2 (a). The lower bound value is given by summing the slot numbers of the last packet of each message: $LB_W = 1 + 2 + 4 + 1 + 3 + 6 = 17$.

4 Properties

In this section, we investigate some properties of our problem, and give two lower bounds on W_S. The first one is for the case $C < N$, and is obtained by disregarding both constraints on rows and on columns, and then by using the minimum waiting time optimal policy of shortest transmission time first.

Let us define a modified traffic matrix D_{LB} of size $C \times \lceil \frac{T}{C} \rceil$. The non-zero entries of D are first placed in a vector v of proper size, and then sorted by non-decreasing values. After this, the entries in v are placed in D_{LB} by column first, and following their place in v: the first entry of v is placed in row 1 and column 1 of D_{LB}, the second in row 2 column 1, and so forth untill the Cth row. This is repeated by considering all entries in v (see Figure 1). Finally, the lower bound is computed as for matrix D' of the case $C = N$, namely by the progressive sum of the non-zero entries in each row i, and then by summing all these row progressive sums (see below).

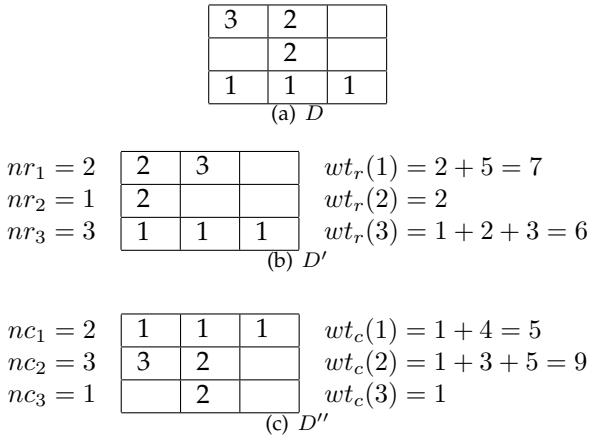


Fig. 2. Lower bound computation in case $C = N$.

A more refined lower bound can be computed in the following way, but only in the special case of un-

3	2	
	2	
1	1	1

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

	1	
1		

1		
	1	
		1

1		
	1	

1		

	1	

	1	

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

1		
	1	
		1

1		
	1	

1		
	1	

	1	
1		

	1	

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

	1	
1		

1		
	1	
		1

1		
	1	

1		
	1	

	1	

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

Fig. 3. 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.

constrained communication, namely when $C = N$. Let *modified by rows traffic matrix* D', an $N \times N$ matrix built by considering each row at time, and rearranging the non-zero entries in non-decreasing order. Similarly, let *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 2, an example of D, D', D'' is shown. The *waiting time of row* i *(column* j*)* $wt_r(i)$ ($wt_c(j)$) is 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 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 all the messages shorter than the pth one, when it would be scheduled after those. An example of $wt_r(i)$ and $wt_c(j)$ computation is given in Figure 2. Now, we can 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. Let us consider, for instance, the traffic matrix shown in Figure 3. It is easy to see that $LB_W = 15$ in this case, but a schedule with W smaller than 18 does not exist. Notice that the lower bound is smaller than that shown in Figure 1 since in this case $C = 3$ and not $C = 2$, like in that figure, even if the traffic matrix is the same. In [17], some properties on the schedules for the problem of minimizing the average *packet* waiting time are given. In particular, it is established that optimal schedules are always of minimum length. This is not necessarily true for the problem of minimizing the average *message* waiting time, considered in this paper (e.g., see Figure 3(b)).

By means of examples, we have shown in Figure 3 the following two properties:

Property 2: A schedule can be optimal even if it is not of minimum length - in Figure 3(b), an optimal schedule is shown which is not of minimum length.

Property 3: A minimum length schedule can be not optimal - in Figure 3(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 shows that our problem is different from that in [17], in spite of its formal similarity.

5 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 7 we show that the outputs of the heuristics are close enough to the optimal solution (on the average).

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 [17].

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 [21]:

Timetable Design: Given

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

Question: We ask for a function

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

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, $1 \leq i \leq n$, $1 \leq j \leq m$;
3) $\sum_{i=1}^{n} f(T_i, C_j, h_k) \leq 1$ for all j and k, $1 \leq j \leq m$, $k \leq p$;
4) $\sum_{j=1}^{m} f(T_i, C_j, h_k) \leq 1$ for all i and k, $1 \leq i \leq n$, $k \leq p$.

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 [21]:

Restricted Timetable Design (RTT):

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

$n = 5, m = 4 \quad R =$

1		1	1	$T_1 = \{1, 2, 3\}$
	1	1		$T_2 = \{1, 3\}$
1			1	$T_3 = \{1, 2\}$
	1		1	$T_4 = \{2, 3\}$
1	1	1		$T_5 = \{1, 2, 3\}$

(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 set the number of available channels in MMWT at least equal to $min(n + 2n_{13}; m + 2n_{13} + 2n_{23})$. Notice that in this proof the number of available channels does not limit the possible simultaneous transmission of different messages, which is only bound by the "same line" constraint. 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 as shown in Figure 5.

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.

6 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.

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

6.1 Greedy (GRE)

This is a very simple heuristic. The schedule is built in this way: all non-zero entries in the traffic matrix are considered in non-decreasing order of their value. A switching matrix is then progressively composed by choosing the entry with minimum value among those not inserted in the matrix yet, and that is in lines both currently exposed (namely, with only zero entries) in such matrix. Eventual ties are arbitrarily broken. After completing the switching matrix, namely when no more entries can be added to it, such a 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 is given in the supplemental file.

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 $O(N^4)$.

6.2 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 choosing the maximum value between the sum of entries with values lower or equal than d_{ij} in row i, and the sum of entries with value lower or equal than d_{ij} in column j.

An example of matrix Q is shown in Figure 8. Matrix Q is then re-computed after each switching matrix generation on the residual traffic matrix. The algorithm description can be obtained by changing *Step 3* and *Step 5* of the previous greedy algorithm, as shown in the supplemental file.

The time complexity of this heuristic is $O(N^4 \log N)$, since the computing of Q is minimized by first sorting the entries of D', and this requires $O(N^2 \log N)$ time.

6.3 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 [22]. For keeping the total waiting time as 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 such a matrix is empty. Pseudocode of this algorithm is shown in the supplemental file.

Max-min matching can be computed in $O(N^{2.5})$ [23], 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.

7 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. The lower bound presented in Section 4 will be used as touchstone: the performance of the heuristics will be represented as percentage increase over the lower bound.

We implemented the algorithms in C language, compiled with *gcc* on a linux machine with Fedora version of the 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 [3], [4]. 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 an m packets message is assigned to switching matrix S^k, all 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 are the details in pseudocode.

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 kth 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 [11], which always produces minimum length schedules;
- the iSLIP algorithm [24];
- a heuristic which builds the schedule in a totally random way.

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

7.1 Optimal algorithm (OPT)

For the sake of completeness, we describe here an exponential-time algorithm to find an optimal 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 a 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.

7.2 BCW algorithm (BCW)

This algorithm [11] always produces minimum length schedules. It has been implemented to show that, in this problem, algorithms designed for producing minimum length schedules can result in performances which are significantly worse than specific 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 first 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 [11].

7.3 iSLIP

iSLIP (Iterative SLIP): this algorithm was proposed by McKeown in [24] and it is an iterative version of the SLIP algorithm. SLIP algorithm is similar to Round Robin Matching algorithm, which is composed of three steps:

- *Request*: each unmatched input sends a request to *every* output for which it has a queued packet;
- *Grant*: if an unmatched output receives any request, it grants the request with highest priority in a fixed round-robin (i.e. in turns) schedule;
- *Accept*: if an input receives grants, it accepts the grant with highest priority.

So, grant/accept *pointers* are increased $moduloN$, N is the number of inputs and outputs. Grant pointers are increased ($moduloN$) only if grants are accepted by the inputs.

iSLIP iterates the previous three steps for a fixed number of times, and at each iteration, only unmatched inputs can join.

iSLIP behaves like SLIP, except for the pointers management: grant pointer to the highest priority entity of the round-robin schedule is increased ($moduloN$) to one location beyond the granted input if and only if the grant will be accepted in the first iteration. Accept pointer to the highest priority entity of the round-robin schedule is increased ($moduloN$) to one location beyond the accepted output only if the input was matched in the first iteration. So pointers are updated only for matches found in the first iteration.

Pseudocode for a sequential version of iSLIP algorithm is reported in the supplemental file.

We remark that iSLIP algorithm has been developed for online scheduling in input-queued switches. We implemented here an offline framed version of this algorithm, namely considering that a number of requests of transmissions are first gathered and then processed.

7.4 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.

8 Simulation results

The simulation tool has been developed to evaluate the goodness of the proposed heuristics. No details about the transmission environment nor the used protocol have been included, since the goal is to study the behaviour of the algorithms with respect to each other. The results have been validated by means of the confidence intervals, which resulted to be always well below the 3% of the average waiting times values (see below), when N is greater than 16; for smaller switch sizes, we obtained larger ratios between the confidence intervals and the

average values. We performed extensive simulations, by tuning several 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 input/output pair;
- the maximum length of the messages, M;
- the traffic matrix $sparsity$, namely the percentage of zero entries.

To make the results more readable, we show them by means of *efficiency* E as performance metric, which is defined as the ratio between the average message waiting time W_S of the schedule produced by a heuristic, and the lower bound LB_W given in Section 4, namely the percentage increase in waiting time that the heuristic has over the lower bound (on the average). So, the smaller is the efficiency value, the better is the performance.

Traffic entries in D matrices have been generated always following two distributions: the *uniform* one, or the *gaussian* one. Since the distributions are countless, we chose two particular ones, just to show whether the algorithms are sensitive to this parameter or not. For the uniform distribution, each traffic entry is given by randomly generating a number between 1 and M, while the number of non-zero entries is established according to traffic matrix $sparsity$ parameter. Also the non-zero entries positions in the matrix are randomly generated.

For the gaussian distribution, we set the mean to M and deviation 1; also in this case, the positions of non-zero entries are randomly generated and the number of zero entries follows the $sparsity$ parameter 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. We show the values for one test case in Table I in the supplemental file, together with all figures referred below.

Is the message waiting time influenced by the switch size? In Figures 1-4, we show the average behavior of the heuristics by varying the switch size N, with uniform distribution and traffic matrix sparsity is equal to 75%, 50%, 25% and 0%, respectively. The same in Figures 5-8, for the gaussian distribution.

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.

Which is the fairer heuristic? In the same Figures, we show also the fairness computed. As fairness metric, we used the well known Jain's index [26], that we report below for the sake of completeness:

$$I_{Jain} = \frac{\left|\sum_{i=1}^{N} x_i\right|^2}{N \sum_{i=1}^{N} x_i^2}$$

where each x_i is computed as the sum of traffic values in the ith row of matrix D, namely representing the total amount of resource allocation for input user i for sending all its messages. All algorithms produce high fairness values, but the proposed Max-Min Matching heuristic gives always the highest values, both in high and low traffic conditions.

Does the channel availability influence the message waiting time? In Figures 9 and 10, we show the behaviour of the algorithms efficiency, by varying the number of channels available for transmission. We considered a switch of size $N = 32$, and we tuned the number of channels C from 2 to 32. The heuristics are not much sensitive to this parameter.

Is the message waiting time affected by the number and the length of messages? In Figure 11, 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. 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 heuristic, instead, non preemptive versions perform worse than preemptive ones, with a loss of about 15% (in terms of efficiency).

In Figure 12, we show the efficiency values obtained by changing the message size. Again, note that the heuristics are not much sensitive to this parameter, too.

How the traffic matrix sparsity affects the message waiting time? Figure 13 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. This happens because when the matrices are sparse it is most likely that the few non-zero entries are in competing positions, so the waiting times of some messages grow. On the converse, when a matrix is totally filled, there is a greater number of scheduling possibilities and then competing entries can be placed in different switching matrices.

How good are the heuristics with respect to the optimal algorithm? In Figures 14 and 15, 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.

Finally, notice that the algorithms performances are not significatively influenced by the distributions used to generate the traffic matrices, in spite of their large difference.

8.1 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);

- 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 of the supplemental file. Notice that MMM heuristic improves its performance with high values of N, leading to less than 1% performance degradation with respect to the lower bound, both in terms of total waiting time values and of schedule lengths. We recall that the AveGapW is computed on the lower bound, and not on the optimal value, and so the performance of the heuristics should be better than the given values. A similar consideration holds for MMM_{NP}, which provides good schedules, even with the non-preemption constraint. So, we conclude that these heuristics give very good sub-optimal solutions to the MMWT problem.

9 Conclusions

In this paper, we studied the problem of minimizing the average message waiting time in a single-hop multichannel system. We shown that this problem is NP-complete, and we proposed and analyzed three fast heuristics. By means of simulations, we obtained the performances of the proposed heuristics, and compared them with other known algorithms. Based on these, we can say 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, since 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 [17]) is unknown. A stimulating research is the MMWT problem in a real-time setting, namely when messages have deadlines to be met.

References

[1] O. Sinnen and A. Sousa, "Communication contention in task scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 503–515, June 2005.

[2] N. McKeown , "The i-SLIP scheduling algorithm for input-queued switches," *IEEE Transactions on Networking*, vol. 7, no. 2, pp. 188–201, April 1999.

[3] 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.

[4] 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.

[5] 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.

[6] H-Y.Wei, S. Ganguly, R. Izmailov, and Z. Haas, "Interference-aware ieee 802.16 wimax mesh networks," Proceedings of *61st IEEE Vehicular Technology Conference, 2005. VTC 2005-Spring* vol. 5, 2005, pp. 3102–3106.

[7] A. Zaki and A. Fapojuwo, "Efficient scheduling algorithms for multi-service multi-slot ofdma networks," Proceedings of *IEEE Wireless Communications and Networking Conference, 2009* vol. 1, April 2009, pp. 1–6.

[8] S. De Vuyst, S. Wittevrongel and H. Bruneel, "Transform-domain analysis of packet delay in network nodes with QoS-aware scheduling," *Lecture Notes in Computer Science*, vol. 5233, pp. 921–950, 2011.

[9] J. Walraevens, D. Fiems, S. Wittevrongel and H. Bruneel, "Calculation of output characteristics of a priproty queue through a busy period analysis," *European Journal of Operational Research*, vol. 198, issue 3, pp. 891–898, November 2009.

[10] 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.

[11] 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.

[12] P. Barcaccia, M. Bonuccelli and M. Di Ianni, "Complexity of minimum length scheduling for precedence constrained messages in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 10, pp. 1090–1102, October 2000.

[13] M. Bonuccelli, F. Martelli, and S. Pelagatti, "Optimal packet scheduling in tree-structured leo satellite clusters," *ACM/Kluwers Mobile Networks and Applications*, vol. 9, no. 4, pp. 289–295, August 2004.

[14] M. Bonuccelli and M. Clo', "Scheduling of real-time messages in optical broadcast-and-select networks," *IEEE Transactions on Networking*, vol. 9, no. 5, pp. 541–552, October 2001.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

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

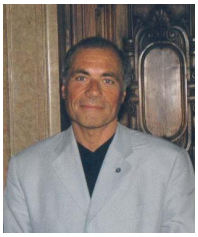
[23] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*. Prentice Hall, 1993.

[24] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches", *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, April 1999.

[25] IEEE Std 802.16-2004, "IEEE standard for local and metropolitan area networks part 16: air interface for fixed broadband wireless access systems", 2004.

[26] R. Jain, D. Chiu and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system", *DEC Technical Report 301*, 1984.

Francesca Martelli Francesca Martelli is a post-doc researcher at the Istituto di Informatica e Telematica (IIT) of the Italian National Research Council (CNR). She got her Laurea and PhD degrees in Computer Science at University of Pisa in 2000 and 2005, respectively. She has been a research associate at the Istituto di Scienza e Tecnologie dell'Informazione (CNR) in Pisa, from 2001 till 2006. From 2006 till 2010 she has been a research associate at the Department of Computer Science of the University of Pisa. She joined the IIT institute in July 2010. Her research interests focus on distributed systems and algorithms, in particular on medium access control problems in wireless networks, such as packet scheduling algorithms. She is interested also in RFID systems, focusing on the tag identification problem. Other research topics are vehicular networks and MIMO networks. She has published several articles on international journals and conferences related to her research topics.



Maurizio A. Bonuccelli Maurizio A. Bonuccelli is Professor of Computer Science at the Dipartimento di Informatica, University of Pisa, Italy. He has been associated with that department since 1982, with the only exception of the years from 1990 until 1994 when he was a Professor of Computer Science at the University of Rome "La Sapienza". During 1981 he took a sabbatical leave at IBM T.J. Watson Res. Center, Yorktown Heights, N.Y., working in the computer communications group, and spent September 1993 at ICSI, Berkeley, CA, with the TENET group. He has been and is involved in the organizing, program and steering committee of several international workshops and conferences. He was International Conference Coordinator of ACM SigMobile. He is in the editorial board of the international journals "Mobile Networks and Applications" (MONET) and "Wireless Communications and Mobile Computing". His field of interest is the design and management of communication, computer networks, and distributed parallel processing systems, with a special emphasis on resource management and MAC protocols for mobile networks.