Timing Reliability for Local Schedulers in Multi-Agent Systems

Davide Calvaresi[1,2], Giuseppe Albanese[3], Mauro Marinoni[1], Fabien Dubosson[2], Paolo Sernani[3], Aldo Franco Dragoni[3], and Michael Schumacher[2]

[1] Scuola Superiore Sant'Anna,Pisa, Italy
[2] University of Applied Sciences Western Switzerland, Sierre, Switzerland
[3] Università Politecnica delle Marche, Ancona, Italy
{d.calvaresi, m.marinoni}@sssup.it,
{p.sernani,a.f.dragoni}@univpm.it, {michael.schumacher,
fabien.dubosson}@hevs.ch

Abstract. In the last decades, the use of Multi-Agent Systems (MAS) resulted in being the most relevant approach to foster the development of systems performing distributed reasoning, automated/autonomous actions, and regulating component interactions in unpredictable and uncertain scenarios. The scientific community provided numerous innovative contributions about resource and task allocation seeking for optimal/sub-optimal solutions. The adoption of MAS in Cyber-Physical Systems (CPS) is producing outstanding results. However, in current MAS, the actual task execution is still delegated to traditional general-purpose scheduling algorithms running within the agent (local scheduler of behaviors). The main consequence is the incapability to enforce compliance with strict timing constraints (i.e., the impossibility of providing any guarantee about the system's behavior in the worst-case scenario). Therefore, the adoption of MAS is hampered, excluding significant application scenarios such as safety-critical environments.
This paper proposes the schedulability analysis of various task-sets, that are feasible using real-time schedulers, on top of traditional general-purpose solutions. In particular, the study of deadline-missing rate occurring in general-purpose setups, evaluated on an agent-based simulator developed on OMNET++, named MAXIM-GPRT, is presented. The obtained results strengthen the motivations for adopting and adapting real-time scheduling mechanisms as the local scheduler within agents.

Keywords: Timing-Reliability, Deadline-Missing Rate, Scheduling Simulation, Multi-Agent Simulator, Agent Local Scheduler

1 Introduction

Cyber-Physical Systems (CPS) can be composed of a multitude of electronic devices with any sort of size. Therefore, they can be considered as the maximal expression of distributed systems pervading humankind's daily living [11]. Continuously interacting with their surrounding, CPS can collect data exploiting ever-increasing types of

distributed sensors. Feedback and data elaborated locally or remotely feed both networks of simple sensors/actuators and intelligent, more complex, distributed entities. The agent-based paradigm is definitely one of the most prominent and promising approaches supporting a broad range of distributed applications. Multi-Agent Systems (MAS) are profusely contributing to domains such as energy [19], manufacturing [16], e-health [7, 8], and telerehabilitation [10]. However, regardless of application scenarios, dimensions, and distribution, the safety of the system and its users is a crucial requirement.

To operate in safety-critical scenarios, the absence of hardware failures and coding errors is not a sufficient guarantee. Hence, a system is considered able to guarantee its correct execution if it can deliberate the right output at or within a given time, even in the worst-case scenario [5]. Hereafter, such a feature is referred as "timing-reliability". It is the fundamental property of the real-time computation systems, i.e., those systems capable to react within precise time constraints to events in the environment where they operate.

Concerning MAS, a multitude of elements operate simultaneously. Individually or collectively, they have to be able to provide primitive mechanisms guaranteeing timing-reliability. In particular, such components are: *(i)* an "intelligent/strategic" layer (i.e. allows single components and the CPS as a whole to achieve their goals), *(ii)* a communication middleware (i.e. to allow the exchange of information and requests among the components of the CPS), and *(iii)* local policies (i.e. schedulers and heuristics enabling each component to execute its tasks). MAS have been employed in automatic, semi-automatic and highly unpredictable and uncertain environments. Nevertheless, mechanisms such as negotiation, communication, and local scheduling have to operate safely in either one. Moreover, current agent-based frameworks cannot yet support the development of a MAS able to guarantee full compliance [11]. In fact, according to the study proposed in [9], most of the solutions composing the state of the art take the execution of the allocated task for granted. Such an assumption is naive and unsustainable if dealing with real safety-critical applications.

Contributions.
This paper studies qualitatively and quantitatively the behaviors of the local schedulers employed in the most relevant agent-based platforms [9] and the most common negotiation protocols. The obtained results have been compared and discussed with respect to real-time scheduling algorithms.

In particular, considering that general-purpose scheduling algorithms neither consider the deadline notion nor can provide any timing guarantee, to enable a performance analysis we have:

(i) Developed an OMNET++ based simulator implementing: a dynamic number of agents, Contract Net (CNET) and Contract Net With Confirmation (CNCP) negotiation protocols, First Come First Served (FCFS), Round Robin (RR), Earliest Deadline First (EDF), and Constant Bandwidth Server (CBS) local schedulers.
(ii) Generated random tasks-sets characterized by parameters computed according to a uniform distribution.
(iii) Analyzed and discussed the obtained outputs.

The paper is organized as follows: Section 2 presents and elaborates the state of the art, Section 3 describes the simulator, Section 4 presents the tests generation and execution. Moreover, it organizes, describes, and discusses the experimental results, and finally Section 5 concludes the paper.

2 State of the art

In the MAS panorama, the notion of *scheduling* applies mainly to mechanisms of task/resources allocation among the agents within one or more platforms. Although such studies provided significant results, they took for granted the task/behavior execution after *allocation* and the compliance with the agreements (or commitment) stipulated during the negotiation phase [6]. Such optimistic assumption has been shown to be unpredictable, thus unacceptable for safety-critical applications [11].

In most of the cases, existing MAS are powered by platforms supporting the development of agent-based systems. Hence, considering the study proposed by Kravari and Bassiliades [17] as common ground, Calvaresi et al. [9] investigated the local schedulers adopted/proposed by the most relevant agent platforms: almost all have implemented at least one local scheduler. NetLogo [22] and Cormas [4] are two exceptions, delegating the development of local schedulers to programmers. In fact, fostering the implementation of custom versions of behavior schedulers should promote a broader set of algorithms. However, despite such flexibility, the dynamics of the customized schedulers can be attributed to classic algorithms. In particular, MaDKit [14], RePast [12], and Swarm [20] implement the FCFS, GAMA [13] and MASON [18] implement a priority scheduler (e.g. SJF-like), Jason implements an RR applied to structured behaviors, and finally JADE implements a non-preemptive RR [21]. The Jason and Jade implementations of RR result in being FCFS of *intentions* [3] in the first case and of *behaviors* in the second. The FCSF and RR scheduling algorithms are two of the most known algorithms and inspired a multitude of variants.

FCFS (also referred as FIFO) executes *tasks* in the exact order of their arrival (according to their position in the ready queue). The absence of preemption or re-ordering in this mechanism allows to classify the FCFS as "the simplest scheduling policy with minimal scheduling overhead".

RR slices the tasks' computing time on the processor in equal time-quantum. Thus, the tasks in the ready queue are cycled to get the CPU. If a running task is completed, the processor directly computes the next one; otherwise, it saves the task status and puts it back in the ready-queue before computing the next one (context switch). RR is mainly appreciated for its fairness, preventing the tasks' starvation. However, these basic mechanisms, as they are, cannot be employed in safety-critical and real-time operating systems because of the long waiting time, significant response time (which has to be recalculated for any new task arrival [24]) and the lack of mechanisms for dealing with strict-timing constraints.

In light of these factors, for both FCFS and RR, the risk of missing deadlines (however neglected) might dramatically increase, thus degrading system performance and compromising its safety and reliability.

Concerning real-time compliant scheduling algorithms[4], differently from FCFS and RR, it is mathematically possible to prove the respect of strict timing constraints by performing the task admission control [5]. The study of the behavior of MAS employing general-purpose algorithms dealing with tasks characterized by deadlines reveals concrete risks and possibilities of failure.

The next section presents the simulator that has been realized to test different setups and measure the recorded *deadline-missing rate*.

3 The Simulator: MAXIM-GPRT

The objective of this study is to evaluate the timing-reliability of the scheduling algorithms at the bottom of currently available agent-based platforms. To perform such an evaluation, a new Multi-Agent System SIMulator for General-Purpose and Real-Time algorithms, named MAXIM-GPRT, has been realized.

MAXIM-GPRT relies on OMNET++ [23], which is a multi-platform component-based framework written in C++, supporting the development of simulators for distributed-like systems, and providing a crucial support for realizing *(i)* basic components (nodes), *(ii)* interactions, and *(iii)* network and communication means. It is extensible, modular, and provides several simulation libraries, integrating a graphical development, run-time environment, and, most importantly, enabling the development of real-time simulations [2].

3.1 MAXIM-GPRT Structure

The simulator is composed of simple modules communicating with each-other by exchanging messages. Coded in C++, such active modules embody the agents. The basic template offers extendable initialization procedure and primitives to handle the message exchange. The modules have been extended to represent the agents' knowledge, desires, behaviors, and connections. Moreover, the modules are organized in a fully connected network, structured using the specific NED (NEtwork-Description) language. For the purposes of this study such NED-coded network involves two main types of agents:

- *Classic agent:* 0 to n agents sharing only a common generic structure, but characterized by diverse and pliable features;
- *Directory Facilitator (DF):* borrowing the notion by the FIPA specifications about agent platforms [1], the DF contains the mapping of the agents and the related services offered. In the current view, it is meant to be 1 per community.

Figure 1 represents the agent composition. In such a way, it is possible to associate a given task-set, scheduler, desires/goals, knowledge, a negotiation protocol (which is the same for the entire community), and any required heuristic to every agent during their initialization phase.

[4] Schedulers able to guarantee that scheduled tasks will meet their timing constraints.

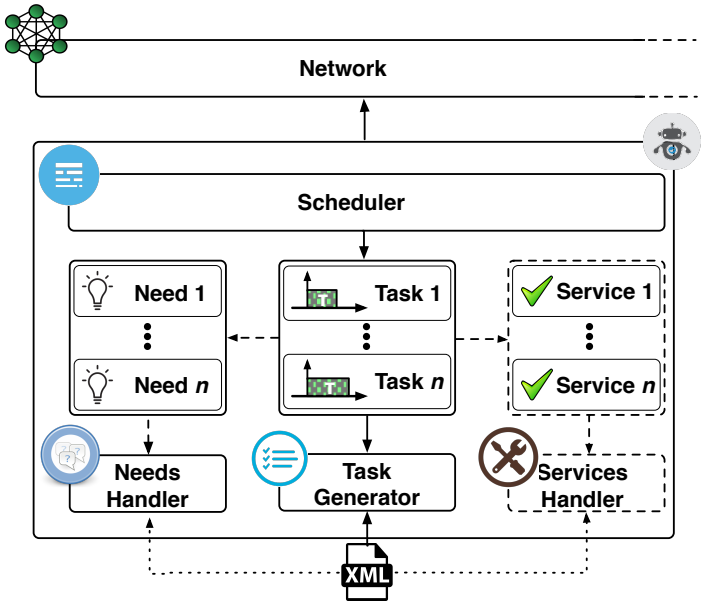


Fig. 1: Agent composition.

In particular, Listing 1 shows the initialization function called only once for each agent at the beginning of the simulation. Lines 3 to 11 load the simulator's settings (e.g., the *maximum dimension* of the *services vector*). Then, a task generator and a needs[5] handler, dealing respectively with a tasks vector and needs vector, are created. At line 15 "If the agent is the DF" follows, the service handler (containing all the services published by the agents at run-time) is created. Otherwise, from line 18 to 21, the initialization function creates all the other objects used by the classical agents: the connections table (mapping the in/out-ports), the heuristic handler (containing the heuristics used in the negotiation phase), and the report files (used to track information such as deadline misses and response time). Finally, from line 22 to 31 the agent's task-set setting, the agent's internal scheduler, the agent's offered services, and its needs are configured (Figure 1).

In the current version of MAXIM-GPRT, the agents can be equipped with:

- *Scheduler:* FCFS, RR, EDF, and CBS;
- *Negotiation prot:* CNET and CNCP;
- *Heuristics:* Considering that the output of the negotiations is not central in this study, seeking simplicity and reproducibility we have: (H1) - if an agent can satisfy a request, it will bid for it, (H2) - the first bidding agent is acknowledged for the task execution.

[5] given tasks which execution is delegated to other agents via negotiation

```
void agent::initialize() {
 if (!ag_sim_has_settings) {
   try {
    EV << "Loading global settings...\n";
     load_ag_sim_settings_from_xml();
   } catch (exception const& ex) {
   cerr << "Error loading the XML configuration file\n";
   exit(EXIT_FAILURE);
   }
} else
 EV << "Global settings already loaded\n";
 ag_settings = new Ag_settings;
 tGen = new TaskGen;
 ag_Needs_Handler = new NeedsHandler;
 if (strcmp("DF", getName()) == 0) {
  df_Service_Handler = new ServiceHandler;
 }
 else {
  ag_settings->set_ag_conn_table(getIndex(), getVectorSize());
  ag_Heurisic_Handler = new Hhandler;
  initialize_xml_report();
  agentMSG *get_ag_tasks_msg = set_ag_tasks();
  scheduleAt(0.0, get_ag_tasks_msg);
  agentMSG *get_ag_sched_msg = set_ag_scheduler();
  scheduleAt(0.0, get_ag_sched_msg);
  agentMSG *set_ag_services_msg = set_ag_services();
  scheduleAt(0.0, set_ag_services_msg);
  agentMSG *set_ag_needs_msg = set_ag_needs();
  scheduleAt(0.0, set_ag_needs_msg);
 }
}
```

Listing 1: Initialize configuration of MAXIM-GPRT

To study *"timing reliability of general-purpose scheduler in MAS"*, it has been decided to employ CNET as the negotiation protocol, focusing on the deadline-missing rate of the two most employed scheduling algorithms. Thus, the two main configurations that have been tested in the several setups presented in Section 4 are (C1): FCFS + CNET, and (C2): RR + CNET.

3.2 MAXIM-GPRT Internal Mechanisms

The agent knowledge is represented by a set of tasks it is able to execute. Moreover, from the same set of tasks, some or all of them can be marked as public, which means that they are "services exposed to other agents in the case of necessity". The execution of such tasks is subject to the negotiation mechanisms.

Whenever a task is released (put into the ready queue), the deadline-check is triggered. When a task is activated (getting the CPU) a check for its completion is triggered.

Depending on the scheduling algorithm, if preemption occurs, the completion check is updated (whilst the deadline-check remains unaltered). These mechanisms enable to monitor the evolution of the system, tracking its behaviors over the time.

Concerning the agents needs, they can be hard-coded or dynamically generated/released at run-time. In the scope of this study, seeking for reproducibility, the needs have been (randomly) generated off-line and then released at run-time (according to the task-sets generations constraints, see Section 4).

To ensure fairness, the needs have been uniformly distributed and generated accordingly to the possibilities offered by the platform in a given setup.

Concerning the negotiation, the CNET has been employed in this study. At the start-up of the platform (or whenever an agent *decides* to make a given service available) the agent communicates to the DF the details about the exposed services which are *published* on the DF table (in the format: agent - service(s)). Exploiting the classic message exchange protocol, whoever needs a service gets the list of possible executors. Then, according to the CNET mechanism, the negotiation takes place, employing the heuristics H1 or H2 (presented in Section 3.1 respectively for selecting the possible contractor and acknowledging the winning bid).

4 Experimental Setup and Results

In the scope of investigating the *timing reliability*, *the deadline-missing rate* recorded by agents employing FCFS and RR as the local scheduler has been analyzed. The study focuses on the execution of periodic tasks. All the generated task-sets are schedulable under the EDF conditions (*Processor Utilization Factor* $U \leq 1$). According to the setups defined in Section 4.1, each task-set ranges regarding agent-utilization and single task-utilization have been fixed. In particular, the values of U_i and C_i have been generated randomly applying a uniform probability distribution. The related periods T_i have been computed according to Equation (1). The task-set generation included the needs generation, which is subject to Equation (1) and to the heuristics H1 and H2.

$$T_i = \frac{C_i}{U_i} \tag{1}$$

By doing so, and considering periodic tasks, all the task sets are feasible (i.e., no deadline are missed), utilizing the EDF real-time scheduler that has been employed as a term of comparison.

Such an algorithm manages the priority according to the absolute deadline (D) of the tasks. Therefore, EDF's ready queue is sorted accordingly, and the task getting the CPU is always the one with the earliest deadline. In the event that a task with a closer deadline than the deadline of the running task is released, a preemption is triggered. According to Horn [15], given a set of *n* independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks is optimal with respect to minimizing the maximum lateness.

It is worth to recall that the *processor utilization factor U* is the fraction of processor time spent in the execution of the task set [5], and it is calculated according to Equation (2). If U is $U > 1$, the schedule is not feasible for any algorithm. If $U \leq 1$,

the schedule is surely feasible for EDF and, depending on the task set features, it might be schedulable for the other algorithms.

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{2}$$

The schedulability test performed by EDF relies on the calculation of U. It is quite easy to be computed, sustainable to be done at run-time, and incremental. For example, if the U of a given running task-set is 0.5, according to Equation 2, by adding a task τ_i with computation time $C_i = 2$ and period $T = 20$ it increases to $U = 0.6$. Checking if a new task can be added at run-time to the task-set has a considerably low computational impact on the CPU and does not require to recompute the whole algorithm.

Missing both the notions of deadline and schedulability test (that can prove off-line the respect of the timing constraints), the behaviors of FCFS and RR have been investigated using the simulator presented in Section 3. The setups employed for such a study are presented in the next section.

4.1 Simulation setups

The parameters characterizing the simulated scenarios employed for both FCFS and RR are: *(i)* number of agents, *(ii)* single agent utilization factor, and *(iii)* single task utilization factor[6]. Concerning the *number of agents* (N^a), the configurations are: $3, 5, 10$ (DF excluded). To better understand the impact of the *agent utilization factor* (U^a) on the deadline-missing rate, three distributions have been tested: low $U^a_l = (0.1 - 0.5)$, medium $U^a_m = (0.6 - 0.8)$, and high $U^a_h = (0.9 - 1)$. Concerning the *utilization factor* (U^τ) *of the single tasks* composing the task-sets, three distribution have been defined: *(i)* low $U^\tau_l = (0.1 - 0.3)$, *(ii)* high $U^\tau_h = (0.4 - 0.6)$, and *(iii)* mixed $U^\tau_x = U^\tau_l \cup U^\tau_h$. Thus, by combining the parameters mentioned above and generating three configurations for each set of parameters, a total amount of 486 task sets has been tested with both FCFS and RR simulations and mathematically proved exploiting the EDF schedulability test.

4.2 Results Presentation

The percentages of deadlines missed obtained in the first run, denoted as (x.x.x.1[7]), is executed for all the nine configurations over 10000 seconds of simulation, are shown in Table 1 for FCFS, and in Table 2 for RR.

Figure 2 shows the number of deadline misses, comparing all the different setups of the first of the analyzed runs (for both FCFS and RR). As expected, the configuration with low utilization for both agents and single tasks does not produce any deadline miss. Similarly, with both agent utilization and single task utilization high, few deadline misses have been recorded. In particular, 8% in configuration 3.2.2.1 (FCFS) and 16%

[6] The single contribution (C_i/T_i) given by a single task to the task-set of the agent executer is referred as single task utilization factor.

[7] The compact format x.x.x.x stands for (N^a) . (U^a) . (U^τ) . (N. of run)

Table 1: Deadlines missed by FCFS in run x.x.x.1 (simulated time 10000 seconds)

		FCFS [%] (ddl-miss/tot-exec)		
N^a	U^a	U_l^τ	U_h^τ	U_x^τ
3	U_l^a	[0%] (0/7987)	[0%] (0/5497)	[0.5%] (24/4175)
	U_m^a	[12%] (1665/13988)	[0%] (0/4662)	[10%] (499/5159)
	U_h^a	[16%] (2665/16319)	[8%] (417/5328)	[9%] (499/5660)
5	U_l^a	[0%] (0/9276)	[0%] (0/10711)	[4%] (195/5502)
	U_m^a	[14%] (2201/15353)	[0%] (0/4756)	[13%] (1091/8500)
	U_h^a	[14%] (2612/18315)	[0%] (0/15450)	[18%] (1677/9335)
10	U_l^a	[0%] (0/12147)	[0%] (0/16810)	[3%] (324/9927)
	U_m^a	[9%] (2781/31366)	[0%] (0/27100)	[5%] (642/13049)
	U_h^a	[13%] (4167/33026)	[0%] (0/22938)	[26%] (3950/15090)

Table 2: Deadlines missed by RR in run x.x.x.1 (simulated time 10000 seconds)

		RR [%] (ddl-miss/tot-exec)		
N^a	U^a	U_l^τ	U_h^τ	U_x^τ
3	U_l^a	[0%] (0/7987)	[0%] (0/5497)	[1%] (48/4175)
	U_m^a	[12%] (1664/13988)	[0%] (0/4662)	[10%] (502/5159)
	U_h^a	[16%] (2665/16319)	[16%] (833/5328)	[10%] (585/5660)
5	U_l^a	[0%] (0/9276)	[0%] (0/10711)	[6%] (328/5502)
	U_m^a	[7%] (1067/15353)	[0%] (0/4756)	[17%] (1434/8500)
	U_h^a	[15%] (2667/18315)	[0%] (0/15450)	[23%] (2165/9335)
10	U_l^a	[0%] (0/12147)	[0%] (0/16810)	[6%] (588/9927)
	U_m^a	[1%] (333/31366)	[0%] (0/27100)	[7%] (970/13049)
	U_h^a	[3%] (1000/33026)	[0%] (0/22938)	[28%] (4239/15090)

in 3.2.2.1 (RR). Such results might be due to the possibility of having harmonic executions[8]. Finally, considering a scenario closer to real-applications (high agent utilization and task-set composed by mixed utilization factors) the occurrence of deadline misses increases dramatically. Hence, with both FCFS and RR, U_x^τ record deadline misses with any tested load (particularly in U_h^a).

For example, Figure 3 presents the snap-shot of the first 200 seconds of the simulation with 5 agents, U_m^a and A_l^τ in FCFS. It is visible, that $task_4$ for agent 2 and $task_6$ for agent 4 systematically miss multiple deadlines. According to the task-sets of such simulation (see Table 3), it means that executing the tasks accepted throughout the negotiation destabilized the internal scheduler, thus causing the generation of the plotted deadline misses.

[8] A task set is said to be harmonic if the tasks have periods that are positive multiple with each other. Given an utilization $U \leq 1$, an harmonic task set is always schedulable and does not produce deadline misses.

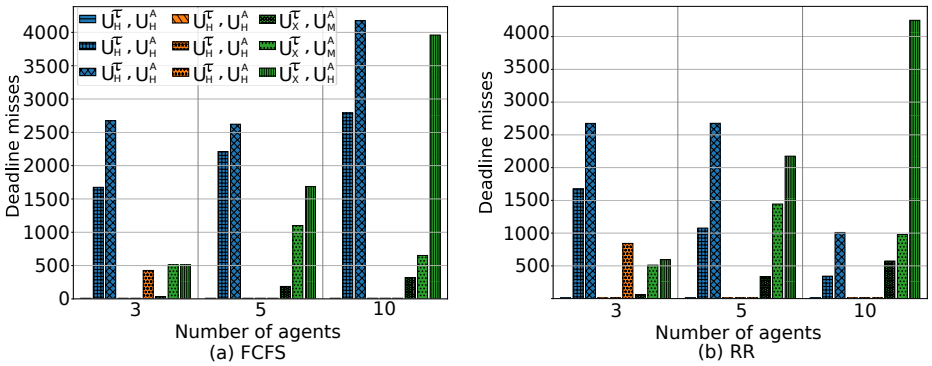


Fig. 2: Overall deadlines missed: run x.x.x.1

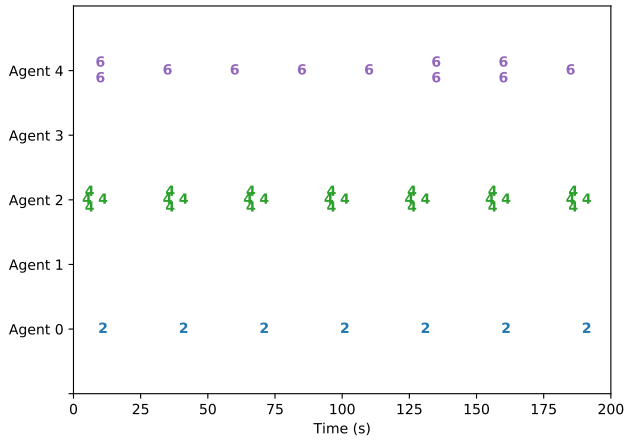


Fig. 3: Deadlines missed for: 5 agents, U_m^a, and A_l^τ [0-200s].

Table 3: Task-set for simulation 5 agents U_m^a and U_l^τ

	$\overline{\lvert A_0 \rvert}$				$\overline{\lvert A_1 \rvert}$				$\overline{\lvert A_2 \rvert}$				$\overline{\lvert A_3 \rvert}$				$\overline{\lvert A_4 \rvert}$			
time	t_i	C_i	T_i	U_i	t_i	C_i	T_i	U_i	t_i	C_i	T_i	U_i	t_i	C_i	T_i	U_i	t_i	C_i	T_i	U_i
	1	6	30	0.2	1	6	30	0.2	1	6	30	0.2	1	6	30	0.2	1	6	30	0.2
	2	2	10	0.2	3	3	15	0.2	4	1	5	0.2	5	4	20	0.2	6	5	25	0.2
t=0				0.4				0.4				0.4				0.4				0.4
t=1	2	2	10	0.2	3	6	15	0.2	4	1	5	0.2	5	4	20	0.2	6	5	25	0.2
	2	2	10	0.2	3	6	15	0.2	4	1	5	0.2	5	4	20	0.2	6	5	25	0.2
				0.8				0.8				0.8				0.8				0.8

Similarly, concerning FCFS, Figure 4c shows per which tasks missed their deadlines agent over the time (first 200 seconds). The analyzed configuration is 10 agents, U_h^a, and U_x^τ (see task-set in Table 4). Such setup shows that long tasks can be delayed due to the interference caused by the execution of smaller tasks.

Table 4: Task-sets for: 10 agents, U_h^a, and U_x^τ.

A_0	t_i	C_i	T_i	Ui	A_1	t_i	C_i	T_i	U_i	A_2	t_i	C_i	T_i	U_i	A_3	t_i	C_i	T_i	U_i	A_4	t_i	C_i	T_i	U_i
	1	6	11	0.54		2	5	9	0.55		3	4	7	0.57		4	10	17	0.58		5	7	12	0.58
(*)	11	6	30	0.2	(*)	12	5	25	0.2	(*)	13	4	20	0.2	(*)	14	10	50	0.2	(*)	15	7	35	0.2
A_5	t_i	C_i	T_i	U_i	A_6	t_i	C_i	T_i	U_i	A_7	t_i	C_i	T_i	U_i	A_8	t_i	C_i	T_i	U_i	A_9	t_i	C_i	T_i	U_i
	6	5	9	0.55		7	11	19	0.57		8	4	7	0.57		9	13	22	0.59		10	12	21	0.55
(*)	16	5	25	0.2	(*)	17	11	55	0.2	(*)	18	4	20	0.2	(*)	19	13	65	0.2	(*)	20	12	60	0.2

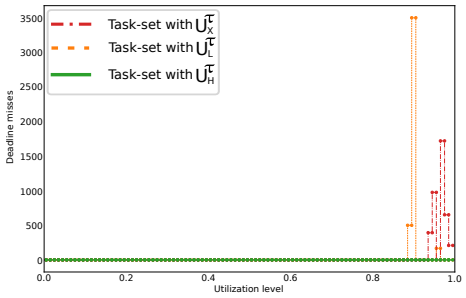
Aggregating the output obtained from all the simulations executed with 10 agents, Figure 4a plots the amount of deadlines missed by the task-set with U_l^τ, U_h^τ, and U_x^τ with respect to the variation of U^a obtained with FCFS.

Except for the task-sets with U_h^τ which do not miss any deadline, as mentioned at the beginning of Section 4.2, the task-sets with U_l^τ have a higher miss ratio with respect to the task-sets with U_x^τ.

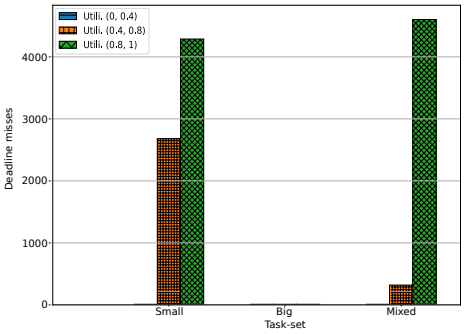
This behaviour can be explained considering that to reach a utilization factor of about 0.8, many small tasks are required (many more with respect to U_h^τ and U_x^τ), thus increasing the probability of competing for the processor due to their different execution frequency.

Nevertheless, for U_h^a mixed task-sets record more deadline-misses with respect to the others, since for higher U, it is possible to allocate more combinations of tasks having U_l^τ and U_h^τ. Figure 4d plots a linear trend of the deadline-missed in the first 1200 seconds for the setup composed by 10 agents with U_h^a and U_x^τ.

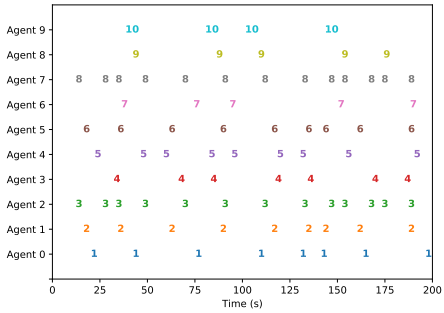
Exploiting the same data aggregated to realize Figure 4a (i.e., deadline missed in all the setups with 10 agents), the distribution of the deadline missed grouped for U_l^a, U_m^a, and U_h^a over U_l^τ, U_h^τ, and U_x^τ is shown (see Figure 4b). Thanks to this visualization, it is possible to emphasize the variation of the deadline-misses by the types of task-sets over the distribution of the U^a. In particular, it is noticeable that the gap between the deadlines missed by U_l^τ and U_x^τ with U_m^a is considerably greater than the gap the same U_l^τ and U_x^τ counted with U_h^a.



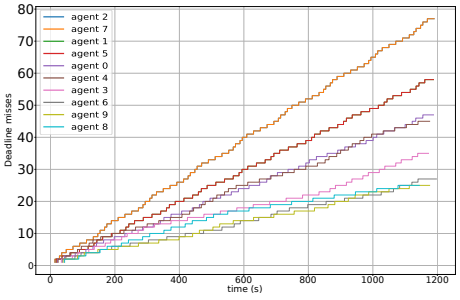
(a) Deadlines missed for 10 agents over U_h^a per U_l^τ, U_h^τ, and U_x^τ.



(b) Deadlines missed for 10 agents over U_l^τ, U_h^τ, and U_x^τ per U^a.



(c) Deadline misses for 10 agents, U_h^a, and U_x^τ [0-200s].



(d) Deadline misses for 10 agents, U_h^a, and U_x^τ [0-1200s].

Fig. 4: Results of the deadline miss analysis.

4.3 Discussion

Analyzing the results provided in Section 4, it is possible to understand that: *(i)* there is no correlation between FCFS and RR's behaviors, and *(ii)* in both FCFS and RR there is a tight connection between the features of the task-sets and the performance of the schedulers. Given this strong dependency, and the impossibility of providing any off/on-line guarantee, the employment of such scheduling algorithms makes existing MAS platforms non suitable for safety-critical applications. Nevertheless, such a variability of behaviors could be tolerated in soft real-time systems, which however would be forced to be over-dimensioned and empirically tested (when possible) in every expected scenario.

Concerning the case of periodic tasks, EDF outperforms considerably FCFS and RR, guaranteeing no deadline misses in all the tested setups. However, it still cannot fully suite the MAS requirements. Hence, recalling that MAS can also negotiate the execution of *one-shot* (aperiodic/sporadic) tasks, it emphasizes that requirements such as *(i)* having mechanisms to handle aperiodic requests (major outcome of sporadic and un-

predictable negotiations) and *(ii)* guaranteeing isolation among tasks (in real-case scenarios, the tasks' computational time cannot always be considered ideal and be trusted by default), cannot be met solely by EDF.

Thus, to deal with the dynamic tasks' activations and arrival times unknown a priori, the Constant Bandwidth Server (CBS) [5] is considered eligible. Moreover, it can deal with dynamic admission tests (whenever it is required to add a new task to the system) ensuring isolation among the tasks (thanks to an efficient bandwidth reservation strategy), and finally, being based on EDF, it maintains the same advantages mentioned above.

Table 5 summarizes its most characterizing features together with those of the schedulers analyzed and discussed in the paper.

Table 5: Improvements required for Local Scheduler.

FCFS	RR	EDF	CBS	Features
☹	☹	☺	☺	No deadline missed for $U \leq 1$
☹	☹	☺	☺	Utilization based acceptance test
☹	☹	☹	☺	Providing schedulability test for aperiodic request
☹	☹	☹	☺	Isolation between sporadic and periodic tasks[9]
☹	☹	☹	☺	Server support and admission test

5 Conclusions

The proposed study analyzes the deadline miss ratio of general-purpose scheduling algorithms employed by the most used multi-agent platforms. To investigate the so-called timing-reliability in MAS, a MAS simulator named MAXIM-GPRT has been developed. Being able of tuning parameters such as agent utilization factor, single task utilization factor, and task-set composition revealed to be a crucial support for the performance analysis.

This study produced evidence eliciting that the employment of MAS in scenarios demanding mandatory compliance with strict-timing constraints is not safe yet. Thus, adoption and adaption of real-time scheduling models in MAS are enforced.

The ongoing work consists in extending the presented study to other task models such as periodic in an interval, aperiodic, and sporadic tasks.

References

1. Foundation for Intelligent Physical Agents Standard. http://www.fipa.org/specs/fipa00023/SC00023K.html, [Accessed 2017-09-24]
2. Albanese, G., Calvaresi, D., Sernani, P., Dubosson, F., Dragoni, A., Schumacher, M.: MAXIM-GPRT: A simulator of local schedulers, negotiations, and communication for multi-agent systems in general-purpose and real-time scenarios. In: Proceedings of the 16th International Conference, PAAMS (2018), https://doi.org/10.1007/978-3-319-94580-4_23

[9] only between the sub-set of the tasks handled by the server and the periodic task-set

3. Bordini, R.H., Hübner, J.F.: Bdi agent programming in agentspeak using jason. In: Proceedings of the 6th international conference on Computational Logic in Multi-Agent Systems (2005)
4. Bousquet, F., Bakam, ., Proton, H., Le Page, C.: Cormas: common-pool resources and multi-agent systems. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. pp. 826–837. Springer (1998)
5. Buttazzo, G.: Hard real-time computing systems: predictable scheduling algorithms and applications, vol. 24. Springer Science & Business Media (2011)
6. Calvaresi, D., Appoggetti, K., Lustrissimini, L., Marinoni, M., Sernani, P., Dragoni, A.F., Schumacher, M.: Multi-agent systems negotiation protocols for cyber-physical systems: Results from a systematic literature review. In: Proceedings of 10th International conference on agents and artificial intelligence (2018)
7. Calvaresi, D., Cesarini, D., Sernani, P., Marinoni, M., Dragoni, A., Sturm, A.: Exploring the ambient assisted living domain: a systematic review. Journal of Ambient Intelligence and Humanized Computing pp. 1–19 (2016)
8. Calvaresi, D., Claudi, A., Dragoni, A., Yu, E., Accattoli, D., Sernani, P.: A goal-oriented requirements engineering approach for the ambient assisted living domain. In: Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments. pp. 20:1–20:4. PETRA '14 (2014), http://doi.acm.org/10.1145/2674396.2674416
9. Calvaresi, D., Marinoni, M., Lustrissimini, L., Appoggetti, K., Sernani, P., Dragoni, A.F., Schumacher, M., Buttazzo, G.: Local scheduling in multi-agent systems: getting ready for safety-critical scenarios. In: Proceedings of 15th European Conference on Multi-Agent Systems (2017)
10. Calvaresi, D., Schumacher, M., Marinoni, M., Hilfiker, R., Dragoni, A., Buttazzo, G.: Agent-based systems for telerehabilitation: strengths, limitations and future challenges. In: proceedings of X Workshop on Agents Applied in Health Care (2017)
11. Calvaresi, D., Marinoni, M., Sturm, A., Schumacher, M., Buttazzo, G.: The challenge of real-time multi-agent systems for enabling iot and cps. in proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'17) (Aug 2017)
12. Collier, N.: Repast: An extensible framework for agent simulation. The University of Chicagos Social Science Research 36, 2003 (2003)
13. Grignard, A., Taillandier, P., Gaudou, B., Vo, D.A., Huynh, N.Q., Drogoul, A.: Gama 1.6: Advancing the art of complex agent-based modeling and simulation. In: International Conference on Principles and Practice of Multi-Agent Systems. pp. 117–131. Springer (2013)
14. Gutknecht, O., Ferber, J.: The madkit agent platform architecture. In: Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents. pp. 48–55. Springer (2000)
15. Horn, W.: Some simple scheduling algorithms. Naval Research Logistics pp. 177–185 (1974)
16. Hsieh, F.: Modeling and control of holonic manufacturing systems based on extended contract net protocol. In: Proceedings of the American Control Conference. vol. 6, pp. 5037–5042 (2002)
17. Kravari, K., Bassiliades, N.: A survey of agent platforms. Journal of Artificial Societies and Social Simulation 18(1), 11 (2015)
18. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K.: Mason: A new multi-agent simulation toolkit. In: Proceedings of the 2004 swarmfest workshop. pp. 316–327 (2004)
19. McArthur, S.D.J., Davidson, E.M., Catterson, V.M., Dimeas, A.L., Hatziargyriou, N.D., Ponci, F., Funabashi, T.: Multi-agent systems for power engineering applications 2014;part i: Concepts, approaches, and technical challenges. IEEE Transactions on Power Systems pp. 1743–1752 (2007)
20. Minar, N., Burkhart, R., Langton, C., Askenazi, M., et al.: The swarm simulation system: A toolkit for building multi-agent simulations (1996)

21. TILab: JADE Manual. http://jade.tilab.com/doc/programmersguide.pdf, [Accessed may'17]
22. Tisue, S., Wilensky, U.: Netlogo: Design and implementation of a multi-agent modeling environment. In: Proceedings of agent. vol. 2004, pp. 7–9 (2004)
23. Varga, A.: Omnet++. Modeling and Tools for Network Simulation pp. 35–59 (2010)
24. Yaashuwanth, C., Ramesh, R.: Intelligent time slice for round robin in real time operating systems. IJRRAS 2(2), 126–131 (2010)