The Challenge of Real-Time Multi-Agent Systems for Enabling IoT and CPS

Davide Calvaresi
Scuola Superiore Sant'Anna
Pisa, Italy
University of Applied Sciences
Western Switzerland
Sierre, Switzerland
d.calvaresi@sssup.it

Mauro Marinoni
Scuola Superiore Sant'Anna
Pisa, Italy
m.marinoni@sssup.it

Arnon Sturm
Ben-Gurion University
of the Negev
Beer Sheva, Israel
sturm@bgu.ac.il

Michael Schumacher
University of Applied Sciences
Western Switzerland
Sierre, Switzerland
michael.schumacher@hevs.ch

Giorgio Buttazzo
Scuola Superiore Sant'Anna
Pisa, Italy
g.buttazzo@sssup.it

ABSTRACT

Techniques originating from the Internet of Things (IoT) and Cyber-Physical Systems (CPS) areas have extensively been applied to develop intelligent and pervasive systems such as assistive monitoring, feedback in telerehabilitation, energy management, and negotiation. Those application domains particularly include three major characteristics: *intelligence*, *autonomy* and *real-time behavior*. Multi-Agent Systems (MAS) are one of the major technological paradigms that are used to implement such systems. However, they mainly address the first two characteristics, but miss to comply with strict timing constraints. The timing compliance is crucial for safety-critical applications operating in domains such as healthcare and automotive. The main reasons for this lack of real-time satisfiability in MAS originate from current theories, standards, and technological implementations. In particular, internal agent schedulers, communication middlewares, and negotiation protocols have been identified as co-factors inhibiting the real-time compliance. This paper provides an analysis of such MAS components and pave the road for achieving the MAS compliance with strict timing constraints, thus fostering reliability and predictability.

CCS Concepts

•**Computing methodologies** → *Distributed artificial intelligence; Intelligent agents;*

Keywords

multi-agent systems, IoT, CPS, real-time systems, real-time multi-agent systems, MAS negotiation timing compliant

1. INTRODUCTION

Technological revolutions deeply changed customs within society, in which human beings are irrevocably coupled with uncountable interconnected electronic devices and their cyber models. This process is still ongoing, providing new application scenarios, while constantly raising new scientific challenges included in the research domains of Internet of Things (IoT) and Cyber-Physical Systems (CPS).

The application scenarios include the domains of assisted living [15, 16], e-health and telerehabilitation [14, 20, 17], smart environments [41, 53], manufacturing [4, 25], and automotive [8]. Indeed, by employing such systems, both users and industries are gaining a broad range of benefits with respect to their needs that include security, comfort, performance enhancement, and low cost. The primary goal is enabling users, applications and machines to understand and interact with their surrounding environment. Facing the users' basic needs, such as water provisioning [33], assistive monitoring [15], communication [16], and mobility [38, 39], different communities (e.g., intelligent and embedded) debate to support their views, approaches, and technologies. Unfortunately, such solutions are mostly technically oriented, whereas the actual needs of a specific application domain and its users are neglected [15].

Various programming paradigms stand behind such systems. As a promising programming paradigm for the application domains mentioned above, Multi-Agent Systems (MAS) [48] gained a significant attention [4, 52]. For example, in the domain of Ambient Assisted Living, MAS has been marked as the most used paradigm [15]. Hence, for both IoT and CPS solutions, MAS seem to satisfy the expected needs while meeting the technical requirements. Leitao et al. [31], for example, developed a manufacturing control system which faced the essential challenge of weaving intelligence, robustness, and adaptation to the environment changes and disturbances. The introduction of multi-agent systems and holonic manufacturing systems paradigms allowed addressing these requirements, bringing the advantages of modularity, decentralization, autonomy, scalability and reusability. Nevertheless, despite the current state of the art, some features that are claimed and presented as ef-

WI '17, August 23-26, 2017, Leipzig, Germany

DOI: http://dx.doi.org/10.1145/3106426.3106518

fective, cannot be matched or guaranteed due to conceptual and technical limitations[17].

Considering the current MAS supporting IoT and CPS solutions, this paper aims at addressing one of the most critical issues, which is *the incapability of MAS to address strict timing constraints*. Associating MAS with real-time related services/tasks in IoT and CPS solutions, represents a fascinating cross-domain. In particular, real-time constraints appear in MAS tasks such as manufacturing process information sharing [4], monitoring [41], and information diffusion and negotiation [33]. If a failure arises and systems are not real-time compliant, the consequences can be irreversible. In this paper we elaborate on the deployment of IoT and CPS solutions and related application domains. We further emphasize the need for real-time compliance and point out the issues that are required to make it possible. Finally, we propose metrics to measure and evaluate current and envisionable contributions proposed to achieve MAS real-time compliance.

The paper is organized as follows. Section 2 discusses the challenges and interactions among MAS, IoT, and CPS. Section 3 describes the strict timing constraints compliance inherited from real-time theory. Moreover, it details the limitations of current MAS in terms of real-time compliance. Section 4 presents the envisioned improvements characterizing the proposed solutions and Section 5 elaborates a simple example showing some improvements with respect to traditional approaches. Finally, Section 6 concludes the paper and presents the ongoing work and some future steps.

2. MAS IN IOT AND CPS: CONTRIBUTIONS AND CHALLENGES

"Internet of Things" recalls a network-oriented vision while focusing on common and heterogeneous objects/devices uniquely addressable. Therefore, it can be integrated into a huge and ever growing network wrapped by several frameworks interacting with the related communication standards.

Figure 1(a), adapted from Atzori et al. [5], represents a general overview of the IoT's components. The first contributions have been recorded around 1999 with the development of Radio-Frequency IDentification (RFID) technologies, which later (∼ 2005) evolved in the development of the Wireless Sensors Networks (WSNs). In the same age, the development of cloud computing and low energy communication led towards the smart things concept (∼ 2012), enabling mobile computing, object interconnection, and cooperation [23]. This last step led to advanced techniques of sensor fusion, distributed intelligence and capabilities of sensing/actuating in the surrounding environment, boosting the IoT evolution.

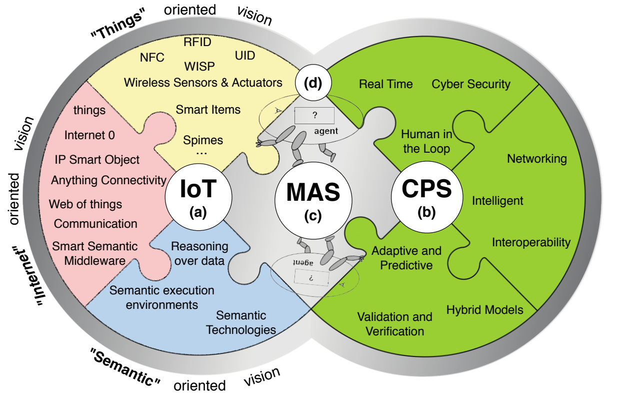

Figure 1: Representations of (a) IoT, (b) CPS, and (c) Intelligent agents (adapted from Russel and Norvig [45]); (d) MAS's overlap supporting IoT and CPS solutions.

Cyber-Physical Systems refer to engineered systems identified by seamless integration of physical components and their cyber models and elements (i.e., computation and communication) [43]. Such a tight integration characterizes the complete system life cycle (from design to run-time phase). CPS extend from minuscule intracorporeal medical devices (e.g., pacemakers) or wearable devices systems (e.g., motion detection and monitoring) to geographically distributed systems (e.g., national power-grids). Cyber-physical systems implicitly involve features such as *adaptability*, *scalability*, *safety*, and *resilience*. Therefore, they stimulated huge interests producing considerable contributions from both artificial intelligence and embedded systems communities. The dynamics of the physical processes must be abstracted from real scenarios, and the resulting models have to be integrated (at design time) while analyzing the performance of the comprehensive cyber-physical system. For example, Biondi et al. [9] proposed an approach for timing analysis of adaptive variable rate (AVR) tasks in power-train applications. They demonstrated that analyzing the entire CPS enables reducing the pessimism in the analysis, thus allowing it to reach a higher CPU utilization and performance while maintaining the timing constraints imposed by the physical system.

The design and implementation of a CPS require a thorough understanding of the application domain, regarding both its users and operating environments. Some wearable systems can be seen as particular cases of CPS, where the human element has to be considered a part of the CPS itself.

A particular example of this class are systems transforming motion-related data into multimodal feedback. Cesarini et al. [19] presents the design and practical implementation of a system providing a real-time acoustic feedback for aquatic-space actions (e.g., performed by swimmers or rehabilitating patients). The system is composed of pressure sensors placed on the swimmer's hands and a small wearable embedded system processing the pressure signals produced by the swimmer motion and providing it in real-time to both swimmer and trainer/therapist. Such a practice, of providing feedback in the form of a functional sound, is called *sonification* [50].

Several solutions rely on elements from IoT combined to those characterizing CPS (see Figure 1(b), adapted from [1]).

Moreover, a subset of both independent and heterogeneous IoT and CPS solutions relies on agent-based frameworks. Such frameworks can be represented as communities of interacting intelligent agents. Aimed at being a human alter ego in its essence and interactions, an intelligent agent can be rationalized as an autonomous entity observing the surrounding environment through sensors and possibly interacting with it using effectors, see Figure 1(c). Self-developed or induced goals (both pre-programmed decision and dynamic ones) drive the agent choices while trying to maximize its performance. Such an intelligent agent is also able to extend/update its knowledge base, thus renewing its plans to achieve the desired goals [45].

The natural abstraction of MAS, in ecological and societal terms, supports the robustness of their mechanisms and behaviors. For example, Zambonelli and Omicini in [52] assert the affinity of *ant foraging* and the *agents' mobility* in finding information within a distributed P2P network, and the similarity of social phenomena like the information propagation in social networks and routing algorithms. Social and natural phenomena with negotiation-based interactions [28] and social conventions [21, 37] have been exploited extensively shaping the multi-agent system paradigm. Moreover, it has also associated physical phenomena like virtual gravitational fields to the orchestration of the overall movements of a vast number of distributed mobile agents/robots [36].

The complexity range of such agents is notably broad. Observing one or more agent communities operating in IoT and CPS scenarios can unveil an apparently unlimited potential. For example, the application domains that received notable contributions are healthcare [52, 54, 11, 24], smart environments (e.g., office, home city [4, 44, 6, 32]), smart cities (e.g., mobility [24, 32], urban safety [52], water distribution [4, 33], transportation [11], and energy [41, 52, 11]), industrial scenarios (e.g., manufacturing [4], workflow and process management [52, 11]), assisted living [15, 16, 44, 6], and telerehabilitation [17].

Related to specific or more general application fields, every (programming) paradigm has its own strengths relying on exposed peculiarities. The MAS paradigm is conceptually elegant and proposes a comprehensive set of features. Investigating primary studies, we identified and collected in Table 1 MAS' characteristics recurring in several studies. Although scenarios and application domains might be different, the authors of the primary studies identified features supported (either partially or fully) by the various adoption of MAS in IoT and CPS. Full support indicates that MAS provide means to satisfy such a feature, while partial support indicates that MAS' contribution, although positive, has been assessed as unable to ensure the complete satisfaction of such a feature.

The proactiveness and the possibility of performing dynamically intelligent behaviors with a high degree of autonomy are the most important features of MAS. Furthermore, MAS resulted in being particularly appreciated in the case of failure handling or resource optimization where required [6]. Finally, although broadly appreciated, MAS autonomy and flexibility still generate minor concerns about possible evolution in undesired behaviors of inferences and plans.

Nevertheless, MAS are increasingly involved in concrete systems, such as the control of physical devices in smart environments (e.g., water provisioning [33]), the energy negotiation, management [53], and systems security [54]. Moreover, in IoT and CPS solutions, the agents have been associated with real-time related services/tasks, representing a fascinating cross-domain class to be analyzed in more depth. For example, in "smart" and other relevant domains, several applications require real-time-like features such as sharing information [33], awareness of environmental changes [11], decision support [33], perception of provided energy [41], information sharing in manufacturer processes [4], security controls [54], and *on time* activities execution in production lines [31]. Such services are receiving increasing scientific attention, and the MAS, if extended with the above-mentioned real-time services, represent a notable overlap among the IoT and CPS systems.

Table 1: *MAS'* feature supporting their adoption in *IoT* and *CPS*

Feature	Contribution	Source
Enable lightweight device coop.	partial	[6]
Increase dependability	partial	[4, 31]
Increase interoperability	partial	[4, 31]
Optimize energy consumption	partial	[44]
Enable repetability	partial	[44, 31]
Facilitate development (various systems' complexity)	partial	[51, 52, 33]
Reducing communication (Agent Migration)	partial	[51, 6]
Facilitate understanding system model	partial	[32]
Enable self-healing	partial	[41]
Handling variability and resources scarcity	partial	[6]
Enabling self-adaption	partial	[52]
Simplify software development/extension	partial	[6]
Ensure robustness	partial	[41]
Facilitate components evolution and reuse	partial	[6]
Face unpredictable scenarios	partial	[52]
Support security (cyber and physical layers)	partial	[54]
Maximization of resources utilization	partial	[4]
Reduce redundancy	partial	[32]
Proactiveness and intelligent behaviors	full	[51, 44, 52] [41, 33, 31]
Ensure Scalability	full	[4, 31]
Reactivity	full	[51, 44, 41]
Social-able	full	[44, 41]
Increase autonomy (e.g.: failures, resources)	full	[4, 51, 44] [41, 33]
Ensure modularity & encapsulation	full	[52, 31]
Support contex awareness	full	[44, 6, 52, 33]
Ensure flexibility	full	[44, 52, 33, 31]
Increase systems integration	full	[4, 44, 31]
Support fault-tolerance	full	[33, 32, 54]
Enable high-level protocols and langs	full	[52]
Ensure reconfigurability	full	[4, 52, 31]

However, current MAS, fail in dealing with real-time properties. Indeed, they typically adopt best-effort approaches, under which the system behavior in worst-case scenarios cannot be handled, nor guaranteed in advance. Ensuring real-time compliance would be a priceless milestone for agent-based solutions. The next section presents the limitations of the current solutions discussing the challenges and the required interventions for having a MAS real-time compliant.

3. MAS & REAL TIME: ANALYSIS, LIMITS AND SOLUTIONS

Medical, industrial, and automotive systems are receiving contributions from a plethora of research fields, each one with its own approach in terms of algorithmic solutions and real-time behavior. Since deriving from very different perspectives, when such worlds need to interact or be merged into comprehensive solutions, the misconceptions existing in different areas may create inconsistencies when they are integrated into a unified solution. A low level of predictability is typically the major consequence when ad-hoc empirical real-time techniques are wrapped into a MAS.

Apparently, the system may operate properly even though all critical time constraints are not verified a priori. For example, when the operating system does not include specific mechanisms for handling real-time tasks and services. Nevertheless, under rare and unpredictable circumstances, the system may collapse without any clear cause. Considering the possibility of critical failures of the MAS operating in the domains presented in Section 2, the consequences of such sporadic failures can be catastrophic, causing physical injuries, environmental damages, and hence financial losses [13].

In addition to classical faults due to code failures, hardware failures, and conceptual errors in the design phase, a real-time software may be subject to timing errors:

Timing errors *may be caused by extra delays introduced by scheduling and synchronization mechanisms, or by misalignments between the "real" time evolving in the environment and the internal system time representation.*

Although the term *"real time"* is associated with the system capability of responding to external stimuli within a bounded amount of time, the common interpretation that *a system is "real-time compliant" if it is able to respond as fast as possible* is not always correct, confusing computational speed with predictability.

Indeed, despite of MAS' complex interactions, often occurring at a time scale of milliseconds, mobile agents could exhibit a real-time behavior if properly controlled. The inability of the current MAS solutions to guarantee strict timing constraints is due to multiple factors addressed below.

3.1 MAS on mobile and embedded devices

MAS frameworks provide generic and extendable functionalities supporting the standardized development of agent-based platforms. Kravari et al. [30] survey MAS frameworks enumerating and detailing the most relevant twenty-four. Most of them run on general-purpose and mobile operating systems (OS) such as (Linux, Mac OS, Windows, or Android), with a few powered by a Java Virtual Machines (JVM) which can claim to be cross-OS. The combination of multi-purpose or mobile OS - MAS as is, cannot guarantee the respect of timing constraints due to "missing" rules and mechanism.

Despite the broad range of compatibility, none of the traditional MAS is meant to run on a proper real-time operating system (RTOS). For example, Calvaresi et al. describe a mobile robot powered by a MAS (coded in JADE and running on a Pandaboard and on an Android-based smartphone) and Erika RTOS (running on a Discovery STM32) [18]. In that project, the robot motion is managed by the Discovery board which is unable to run the MAS (due to limited resources, and its inability to run JVM). Thus, the adopted solution was to wrap the motion' functionality within one of the agents. Such a solution confirms that even the JADE's backbone needs radical changes to match strict real-time constraints. The difficulties stated by Calvaresi et al. revealed several problems addressed in the next sub-sections.

3.2 MAS' elements inadequate for real time

The fundamental elements characterizing a MAS are: *agent internal scheduler*, *communication protocol*, and *negotiation protocol.* Unfortunately, upgrading these elements individually is not enough for providing the expected real-time compliance. For example, Julian et al. [27] provided an extension of a method named *"Message"* for developing a MAS pursing real-time compliance. According to their analysis, the previous method was not sufficient to guarantee the aimed real-time requirements for the following reasons:

- the protocol was operating in a framework that is not meant for facing real-time needs;
- its low-level layer required an ad-hoc design tailored for any specific situation;
- several extensions were required to incorporate all the temporal aspects; and
- diverse criticalities had to be considered.

The proposed methodology introduced concepts such as worst-case execution time (WCET) and schedulability analysis, trying to cope with the overall process of developing a real-time MAS. Although they have foreseen important aspects to be included in such a process, a complete framework matching all the required features is still missing.

3.2.1 Agents' internal scheduler

Current MAS frameworks schedule their different tasks (known as Behaviors) using mainly Round-Robin (RR) and first-come first-served (FCFS) or versions of those. For example, JADE implements a *non-preemptive Round-Robin* scheduler [3]. Unfortunately, such approaches do not deal with timing constraints, which however are crucial in safety-critical scenarios. To enable the mapping with the real-time elements, it is worth analyzing the most used behaviours (e.g., Jade's primitive and composite classes). The *primitive* behaviors are:

- **SimpleBehaviour:** an extendable basic class;
- **CyclicBehaviour:** a behaviour performing actions repeatedly, reactivating itself after its execution is completed. It stays active as long as its agent is alive;
    - **TickerBehaviour:** a periodic behavior which unlike the *CyclicBehaviour* is re-executed after a set time (customized activation period);
- **OneShotBehaviour:** an instance can only be executed once along with its agent life-cycle;
    - **WakerBehaviour:** it allows defining the activation time (delay from the agent life-cycle start);
    - **ReceiverBehaviour:** it is triggered if a timeout expires or a specific type of message is received.

Complex combination of *primitive* behaviors are enabled by *composite* behaviors, such as:

- **ParallelBehaviour:** it enables the parallel execution of children behaviors allowing the definition of the termination conditions: it terminates if *all, n,* or *any* child is completed.
- **SequentialBehaviour:** it executes its children behaviors consecutively and terminates when the last child is terminated.

To exploit the scheduling algorithms inherited from the real-time theory, such behaviors have to be mapped on the real-time tasks models.

3.2.2 Agents' communication middleware

According to their social conception, within and between current MAS, agents interact, communicate, and negotiate activities and resources by exchanging messages.

To understand each other, the definition of common formats and semantics (possibly standard) are necessary. For example, similarly to the other major MAS, JADE is compliant with FIPA standards [2]. Thereby, its agents could interact with any FIPA compliant agent (language and platform independent). The FIPA message structure presents both mandatory (e.g., message type indicated as performative - *request, inform*) and optional (e.g., recipient, sender, ontology) contents. In JADE, messages adhere strictly to the ACL (Agent Communication Language) standard which allows several possibilities for the encoding of the actual contents.

Those contents, packaged as messages, are sent over IP without any mechanism to handle and rule:

- network load and messages status (e.g., possible congestion and delivering time are impossible to be bounded);
- incoming and outgoing messages queue (e.g., agents cannot have an in/out messages awareness);
- broadcasting (e.g., no light mechanism to broadcast simple information such as sensors values).

3.2.3 Agents' negotiation protocol

Communities of agents can achieve mutual agreements, organize activities, optimize efforts and resources pursuing private or common goals, and plans exploiting crucial negotiation mechanisms. The need for a coordination method for a flexible task allocation to multiple problem solvers (nodes/agents), received several contributions by the Artificial Intelligence (AI) community [26]. Among versions conceived and enrolled by the various MAS, the *Contract Net* (CN) developed by Davis and Smith [49] became a part of the FIPA standard [2].

The negotiation protocol is a set of rules governing the interaction between agents that can be *initiators* (who propose the task to be performed and the related boundary conditions or require a specific resource) and *contractors* (who propose themselves as "solvers" replying to the required conditions with a bid) dynamically. The current *CN* rules consist of participants' types (e.g., the negotiators and relevant third parties), negotiation states (e.g., accepting bids, negotiation closed), motivation of the transitions state (e.g., no more bidders, bid accepted), and the possible actions a participant can perform/propose in a particular state (e.g., which can be sent by whom, to whom and when). The protocol employed, the negotiated objects' nature, and the possible operations define the complexity of the model.

Although argumentation and negotiation in MAS can involve fascinating and sophisticated, high-level reasoning, the relevance of the strict connection with the other MAS components such as agent's internal scheduler and communication middleware (both operating at low level) should not be neglected. For example, accepting a task within a negotiation phase impacts the contractor's task-set. Thus, its functional parameters (e.g., workload, utilization factor, acceptance ratio) must be re-evaluated.

Hence, to interoperate with such components under real-time constraints the negotiated activities should be characterized mandatorily by features (e.g., WCET, inter-arrival time, activation time). The current version of the negotiation protocol does not take into consideration such crucial features as well as does not implement mechanisms to bound the negotiation. For example, the *CN* offers only a parameter named *"replyBy"* which (if used) specifies the time frame useful to submit a bid. However, such a parameter has no impact on the rest of the system's mechanisms, thus leading to an unpredictable process.

Along the years, some proposed extensions tried to introduce *"novel"* timing concepts, but unfortunately, missing the big picture (updating elements singularly), resulted in the inability to guarantee MAS' real-time compliance. For example, Qiaoyun et al. proposed to confine the task announcement handled by the initiator imposing a deadline (timeout) [42]. Thus, the period of time to receive bids for the proposed task is limited to an arbitrary interval. This mechanism introduces improvements overcoming some of the limitations inherited from the original protocol (e.g., diverging negotiations).

4. THE PROPOSED SOLUTION

According to the analysis in Section 3.2, to tackle the challenge of realizing an actual real-time MAS the foreseen intervention should be extensive, involving the MAS core elements (*agent's internal scheduler*, *communication middleware*, and *negotiation protocol*) simultaneously and coherently. Indeed, having all the components (both when operating individually and interacting) reliable and real-time compliant, it is possible to envision the overall MAS acquiring the same characteristics.

4.1 Towards a real-time agent's scheduler

In a real-time system, the correct resource allocation to guarantee the timing constraints is based on an analysis that considers the worst-case scenario for the set of tasks under evaluation. Hence, determining the best fitting task models maximizes the agent resource utilization, which is crucial to provide timing guarantees to the behaviors executed by the agents. For example, the *TickerBehaviour* can be mapped on a *periodic task* typical of real-time applications. Moreover, the knowledge about external activities and incoming packets (i.e., minimum inter-arrival) allows managing the *ReceiverBehaviour* using the *sporadic task* model [13]. According to the involved task models and the various required real-time constraints, several scheduling algorithms can be considered for behaviors' management. Considering a scenario exclusively involving of periodic and sporadic tasks, the scheduling can be performed using well-known algorithms like Rate Monotonic (RM) [34] or Earliest Deadline First (EDF) [13] (depending on specific requirements). To handle less predictable behaviors like *OneShotBehaviour*, the real-time approach based on aperiodic servers (e.g., Sporadic Server (SS), Total Bandwidth Server (TBS), and Constant Bandwidth Server (CBS) [13])) is the most suitable. Hence, aperiodic servers are in charge of managing incoming requests bounding the maximum computation bandwidth provided to each specific task. Such a solution provides both isolation among tasks and reduces the pessimism in the timing analysis. For example, if a *ReceiverBehaviour* is triggered more than expected by external events it could create starvation in other behaviors when scheduled with scheduling algorithms currently used in MAS, but will be the only one suffering overload if handled by a CBS. The use of semaphoric primitives to protect critical sections and shared variables could introduce unbounded delays in the response time. Specific protocols such as *priority inheritance* [47] and *stack resources policy* [7] should be used to avoid unbounded blocking and meet timing constraints. The use of dedicated real-time techniques is mandatory for any other aspect that has to be addressed, like overload management, execution on multi-core platforms, and energy management. For example, Buttazzo has shown that a naive use of power-aware features can jeopardize the real-time guarantees [12].

4.2 Towards a real-time communication middleware

The challenge of providing temporal guarantees to the communication layer is arduous due to the additional constraints that require further considerations. The most significant one concerns the protocols interoperability and the impossibility to bound network reliability. Depending on the flexibility in terms of requirements, predictability can be improved at different levels of the ISO/OSI stack. The real-time community has spent a considerable effort to propose more deterministic solutions for the Physical and the Data Link layers. Most of them use a slot-based approach, which allows avoiding the contention of the media and retransmissions due to collisions, therefore enhancing the predictability.

For example, Crenshaw et al. [22] proposed the RI-EDF protocol that uses the EDF algorithm to schedule the allocation of transmission slots in a wired or wireless sensor networks. A similar approach has been proposed by Ma et al. [35] for Wi-Fi access points with energy and time constraints. At the transport layer, some protocols have been proposed to improve predictably of transmission times, along with providing a more scalable paradigm for data interchange. An example is the Data Distribution Service (DDS) [40] which implements a distribute publish-subscribe communication mechanism able to manage the quality of service for the transmitted packets. Overcoming the limitations of communication middlewares employed by current MAS can be achieved by refining the aforementioned mechanisms to the real-time MAS requirements and purposes, thus also enabling the reliability of the other components.

4.3 Towards a real-time negotiation protocol

The negotiation process is a key mechanism within a MAS, which also relies on agents' internal scheduler and communication middleware. Both *initiators* and *contractors* must perform predefined steps to take part in the negotiation. These steps involving both parties require to be extended and modified establishing some mechanisms to achieve and guarantee the real-time compliance. Thus, dilemmas such as *"the Eager Bidder Problem"* [46] can be avoided or bounded with a certain level of predictability. For example, enriching the information available to the agents regarding each other's state, specific performance indexes, and their recent trends can enhance the agents' characterization. Thus, supporting the initiator in a better definition and organization of possible bidders for its announcements. A better characterization of the possible *contractors* reduces and optimizes network load and tasks allocation. Regarding the *contractor*, new internal mechanisms are required to handle the aforementioned parameters. This combination of new mechanisms and parameters will enable the timing constraints compliance. Indeed, it will generate an agent self-awareness about its utilization factors, task-set, and tasks' deadline, exploiting as much as possible the flexibility provided by the related real-time scheduler. Thus, both tasks already accepted and running on the agent (contractor) and tasks under bid will have strictly guaranteed the undertaken and promised response time.

5. IMPROVEMENTS OF THE PROPOSED SOLUTION: A PRACTICAL EXAMPLE

This section exploits a simple example to present the behaviors produced by classical approaches such as *CNET* [2] and CNCP [29] alongside the behaviors obtained by combining the possible improvements presented in Section 4. Finally, limitations and advantages are summarized.

Figure 2(a) depicts four interacting agents. In particular, agent A negotiates with agents B and C the execution of a task τ_k, while the agent D negotiates the execution of a task τ_y with B.

Figure 2(b) shows the temporal sequence of events occurring in the four agents' interaction. The requests for τ_k, sent from A at t_0, are received at t_1 by B, and at t_2 by C. At t_3, D sends its request for τ_y to B. Such a request for τ_y is received at t_7 by B, hence before having received either a confirmation or a reject to its bid for τ_k proposed to A. If requests are negotiated with *CNET*, B would reject the request of D since it is under bid and no guarantee or further bids can be provided yet. This can result in a potentially huge loss of computational capacity being committed to a not yet awarded task [29]. If the agent "under bid" receives a reject, this means that it has refused, for unfounded reasons, all the requests received in the interval between the moment it submitted its bid and the moment it received the notification. This approach would reduce predictability because task acceptance does not solely depend on the total workload but also on the requests order.

Instead, by adopting the negotiation mechanism proposed in [29], B would bid to both A and D (since it has the "capability" to perform τ_k), waiting for a second phase to actually commit to the initiator which first awarded the bid (sending a reject to the other agents). Such a solution overcomes the limitations introduced by CNET [2]. However, the mechanism of bidding to all the initiators, followed by a withdrawal in a second stage, introduces other limitations such as *unreliability* and *unpredictability.* This involves refusing all the initiators that the contractor bidded, except the one that awarded the task first, thus destabilizing the system.

However, considering the approach proposed in Section 4 the drawbacks of these two negotiation protocols could be overcome. Employing one of the schedulers proposed in Section 4.1 allows to enforce timing constraints, relying on its acceptability and schedulability tests, and policies which play a crucial role on the mechanism discussed in Section 4.3. Although communication delays cannot be neglected, according to Section 4.2, it is possible to assume them as bounded (referred as δ). Thus, regarding the current example we have:

At time t_0, agent A sends a bid to both B and C. Each one receives the request after a transmission delay, performs

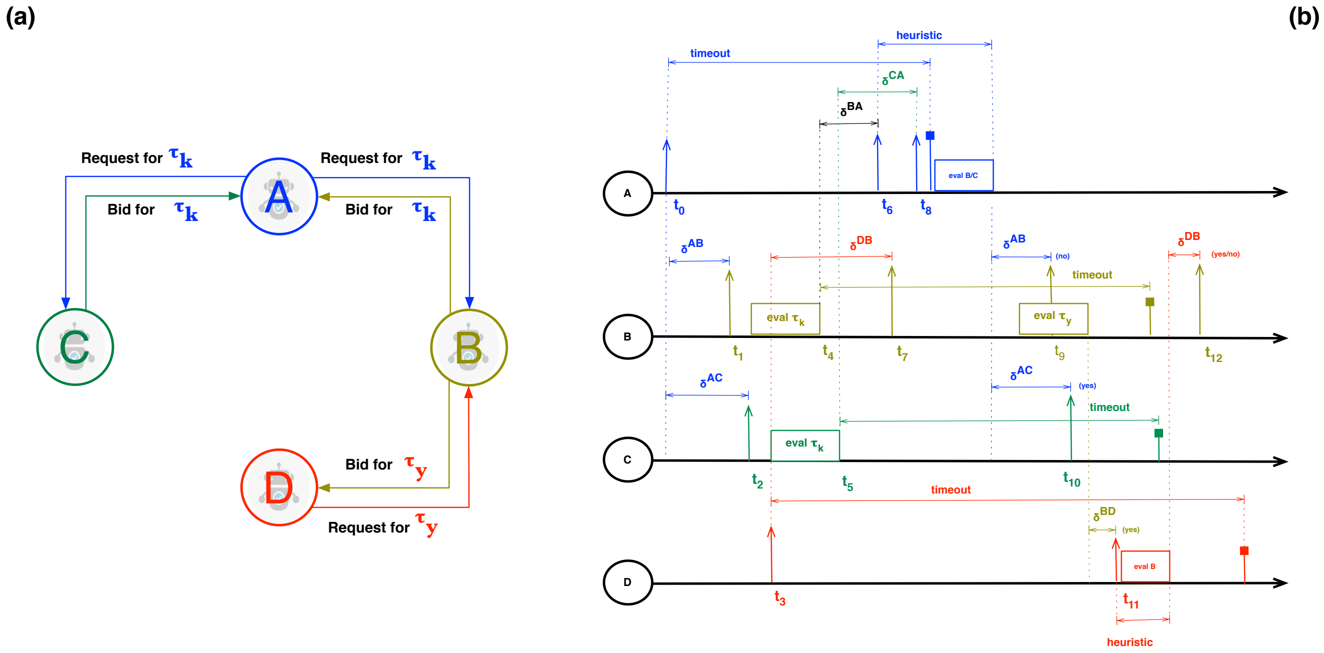

Figure 2: (a) Agents interactions; (b) Schedule of agents A,B,C, and D negotiating tasks τ_k and τ_y.

an acceptance test based on its actual workload (eval τ_k), and replies with the result of the test back to agent A respectively at t_4 and t_5. The test checks if the requested task can be performed on the node within its deadline. This would be done without jeopardizing the time constraints of the current schedule, recalling that the duration of each step can be bounded by an accurate selection of protocols and scheduling algorithms, as described in Section 4.1. At time t_7, B performs the acceptance test on the request from D considering both its current workload and its *potential* one. Since the confirmation of the bid for τ_k from A has not been received yet, the workload due to task τ_k is considered only as potential. In this way, B can notify if it can safely accept task τ_y (i.e., it can execute both τ_k and τ_y before the requested deadline) or if the requested task is already incompatible even with the current workload.

This simple example already highlights how a negotiation mechanism relying on concepts presented in Section 4 can ensure:

(i) the increment of task acceptance ratio and utilization factor with respect to [2], both with respect to the whole system and to the single agents;

(ii) the correct execution of previously accepted tasks. Hence, new tasks overloading the system are not accepted, ensuring the functionalities of the already running task sets, and gaining an increased reliability with respect to [29]; and

(iii) the drastic reduction of the network load with respect to [29].

6. CONCLUSIONS AND ONGOING WORK

This paper presented the need for comprehensive solutions empowering the peculiar feature of both IoT and CPS agent-based systems. Highlighting the still unsatisfied requirement of respecting strict timing constraints, the inadequate components of current agent-based solutions have been detailed. To address this challenge, we mapped the MAS key elements to the real-time aspects. This process requires several interventions in terms of theoretical contributions and practical development of new mechanisms. As a first step, we plan to develop a simulator enabling the evaluation of the introduced improvements towards the real-time compliance. This would allow us to understand and quantify the effects of the changes progressively performed on MAS pillars.

The proposed enhancements regarding the scheduler, the communication middleware, and the negotiation protocol will be evaluated and compared with current solutions from state of the art. Usually, MAS operate in a highly broad set of scenarios. Thus, the simulations will have to cover a full suite of situations varying the number of initiators, tasks to be announced, and contractors. Moreover, for every combination of those above, the characteristics of every component will be varied in a broad range of values. The critical scenarios that will arise (e.g., overload and overhead) will be deeply analyzed (for each algorithm and protocol) to understand the underlying causes and the effects on both single agents and the whole system).

According to Bozdag et al. [10], relevant metrics to evaluate the obtained results can be *responsiveness, resources utilization*, and *load distribution*. Furthermore, it will be considered the employment of the *overload-related indexes* proposed by Buttazzo [13].

Analyzing the *responsiveness* aims to measure the highest *response time* of a single task or a task-set. Employing real-time techniques it is possible to calculate reliable system's bounds. In the case of periodic tasks, techniques such as Response Time Analysis (RTA) will be employed, whereas in the case of sporadic tasks, indexes such as lateness (how far in advance of its deadline a task terminates its execution) will characterize the analyzed components.

Although in the real-time theory the *resources utilization* is worst-case oriented, this factor allows understanding and maximizing of resources utilization while respecting the timing constraints. In some cases, employing real-time algorithms could reveal lower utilization with respect to naive over-provisioning techniques, which however, do not guarantee the adherence with timing constraints.

Analyzing the *load distribution* enables to understand how balanced the task allocation is with respect to agents capabilities (computational and physical), energy management constraints (e.g., impact on battery lifetime), and fault tolerance policies (e.g., in the case of required redundancies, a wise diversification is needed).

Finally, moving from simply theoretical contributions to actual implementations, the effects of unexpected overhead situations must be taken into account. Thus, possible *overhead* related to *control, communication, scheduling,* and *negotiation* will be analyzed.

7. ACKNOWLEDGMENTS

The authors wish to thank the contribution of the COST Action IC1303 - Architectures, Algorithms and Platforms for Enhanced Living Environments (AAPELE).

8. REFERENCES

[1] Cyber - Physical Systems. http://cyberphysicalsystems.org/. [Accessed 8/1/'17].

[2] Foundation for Intelligent Physical Agents Standard. http://www.fipa.org/. [Accessed 8/1/'17].

[3] JADE - Programmer Manual. http://jade.tilab.com/doc/programmersguide.pdf. [Accessed 12/06/'17].

[4] C. Alexakos and A. Kalogeras. Internet of things integration to a multi agent system based manufacturing environment. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8. IEEE, 2015.

[5] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[6] I. Ayala, M. Amor, and L. Fuentes. Towards a cvl process to develop agents for the iot. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 304–311. Springer, 2014.

[7] T. Baker. A stack-based resource allocation policy for realtime processes. In *Proceedings 11th Real-Time Systems Symposium*, pages 191–200, Dec 1990.

[8] A. Biondi, M. Di Natale, and G. Buttazzo. Response-time analysis for real-time tasks in engine control applications. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, pages 120–129. ACM, 2015.

[9] A. Biondi, M. Di Natale, and G. Buttazzo. Performance-driven design of engine control tasks. In *ACM/IEEE 7th International Conference on Cyber-Physical Systems*, pages 1–10. IEEE, 2016.

[10] E. Bozdag. A survey of extensions to the contract net protocol. *Technical report, CiteSeerX-Scientific Literature Digital Library and Search Engine*, 2008.

[11] M. Bujorianu, M. Bujorianu, and H. Barringer. A formal framework for user-centric control of multi-agent cyber-physical systems. 2009.

[12] G. Buttazzo. Achieving scalability in real-time systems. *Computer*, 39(5):54–59, May 2006.

[13] G. Buttazzo. *Hard real-time computing systems: predictable scheduling algorithms and applications*, volume 24. Springer Science & Business Media, 2011.

[14] D. Calvaresi, D. Cesarini, M. Marinoni, P. Buonocunto, S. Bandinelli, and G. Buttazzo. Non-intrusive patient monitoring for supporting general practitioners in following diseases evolution. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 491–501, 2015.

[15] D. Calvaresi, D. Cesarini, P. Sernani, M. Marinoni, A. Dragoni, and A. Sturm. Exploring the ambient assisted living domain: a systematic review. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–19, 2016.

[16] D. Calvaresi, A. Claudi, A. Dragoni, E. Yu, D. Accattoli, and P. Sernani. 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*, page 20, 2014.

[17] D. Calvaresi, M. Schumacher, M. Marinoni, R. Hilfiker, A. Dragoni, and G. Buttazzo. Agent-based systems for telerehabilitation: strengths, limitations and future challenges. In *proceedings of X Workshop on Agents Applied in Health Care*, 2017.

[18] D. Calvaresi, P. Sernani, M. Marinoni, A. Claudi, A. Balsini, A. Dragoni, and G. Buttazzo. A framework based on real-time os and multi-agents for intelligent autonomous robot competitions.

[19] D. Cesarini, D. Calvaresi, C. Farnesi, D. Taddei, S. Frediani, B. E. Ungerechts, and T. Hermann. Mediation: An embedded system for auditory feedback of hand-water interaction while swimming. *Procedia Engineering*, 147:324–329, 2016.

[20] D. Cesarini, D. Calvaresi, M. Marinoni, P. Buonocunto, and G. Buttazzo. Simplifying tele-rehabilitation devices for their practical use in non-clinical environments. In *International Conference on Bioinformatics and Biomedical Engineering*, 2015.

[21] P. Ciancarini, A. Omicini, and F. Zambonelli. Multiagent system engineering: The coordination viewpoint. In *International Workshop on Agent Theories, Architectures, and Languages*, 1999.

[22] T. Crenshaw, A. Tirumala, S. Hoke, and M. Caccamo. A robust implicit access protocol for real-time wireless collaboration. In *17th Euromicro Conference on Real-Time Systems*, pages 177–186. IEEE, 2005.

[23] L. Da Xu, W. He, and S. Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.

[24] O. Gürcan, N. Yakymets, S. Tucci-Piergiovanni, and A. Radermacher. Multi-agent optimization for safety analysis of cyber-physical systems. 2015.

[25] A. Houyou, H. Huth, C. Kloukinas, H. Trsek, and D. Rotondi. Agile manufacturing: General challenges and an iot@ work perspective. In *Proceedings of 2012 IEEE 17th International Conference on Emerging*

Technologies & Factory Automation ETFA, 2012.

[26] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998.

[27] V. Julian and V. Botti. Developing real-time multi-agent systems. *Integrated Computer-Aided Engineering*, 11(2):135–149, 2004.

[28] J. Kephart. Software agents and the route to the information economy. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7207–7213, 2002.

[29] T. Knabe, M. Schillo, and K. Fischer. Improvements to the fipa contract net protocol for performance increase and cascading applications. In *International Workshop for Multi-Agent Interoperability (KI)*, 2002.

[30] K. Kravari and N. Bassiliades. A survey of agent platforms. *Journal of Artificial Societies and Social Simulation*, 2015.

[31] P. Leitão. Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7):979 – 991, 2009. Distributed Control of Production Systems.

[32] J. Lin, S. Sedigh, and A. Miller. A general framework for quantitative modeling of dependability in cyber-physical systems: a proposal for doctoral research. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*, volume 1, pages 668–671. IEEE, 2009.

[33] J. Lin, S. Sedigh, and A. Miller. Modeling cyber-physical systems with semantic agents. In *34th Computer Software and Applications Conference*, 2010.

[34] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[35] J. Ma, S. Kim, and D. Kim. Tame: Time window scheduling of wireless access points for maximum energy efficiency and high throughput. In *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2012.

[36] M. Mamei, F. Zambonelli, and L. Leonardi. Cofields: a physically inspired approach to motion coordination. *IEEE Pervasive Computing*, 3(2):52–61, 2004.

[37] Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and Artificial Intelligence*, 1995.

[38] P. Pagano. *Intelligent Transportation Systems: From Good Practices to Standards.* Taylor & Francis, 2016.

[39] P. Pagano, C. Salvadori, S. Madeo, M. Petracca, S. Bocchino, D. Alessandrelli, A. Azzar, M. Ghibaudi, G. Pellerano, and R. Pelliccia. A middleware of things for supporting distributed vision applications. In *Proceedings of the 1st Workshop on Smart Cameras for Robotic Applications, SCaBot Workshop*, 2012.

[40] G. Pardo-Castellote. Omg data-distribution service: Architectural overview. In *Distributed Computing Systems Workshops*, pages 200–206. IEEE, 2003.

[41] M. Pipattanasomporn, H. Feroze, and S. Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pages 1–8. IEEE, 2009.

[42] L. Qiaoyun, L. Jiandong, D. Dawei, and K. Lishan. An extension of contract net protocol with real time constraints. *Wuhan Univerity Journal of Natural Sciences*, 1996.

[43] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference*, 2010.

[44] M. Roscia, M. Longo, and G. C. Lazaroiu. Smart city by multi-agent systems. In *Renewable Energy Research and Applications*, 2013.

[45] S. Russell, P. Norvig, J. Canny, J. Malik, and D. Edwards. *Artificial intelligence: a modern approach.* Prentice hall Upper Saddle River, 2003.

[46] M. Schillo, C. Kray, and K. Fischer. The eager bidder problem: a fundamental problem of dai and selected solutions. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pages 599–606. ACM, 2002.

[47] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: an approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, Sep 1990.

[48] Y. Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1):51–92, 1993.

[49] R. Smith. Communication and control in problem solver. *IEEE Transactions on computers*, 1980.

[50] B. Ungerechts, D. Cesarini, V. Wiebel, and T. Hermann. Ears drive hands: sonification of liquid effects induced by aquatic space activities contributes to cognitive representation, 2015.

[51] H. Yu, Z. Shen, and C. Leung. From internet of things to internet of agents. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages 1054–1057. IEEE, 2013.

[52] F. Zambonelli and A. Omicini. Challenges and research directions in agent-oriented software engineering. *Autonomous agents and multi-agent systems*, 9(3):253–283, 2004.

[53] P. Zhao, S. Suryanarayanan, and M. G. Simoes. An energy management system for building structures using a multi-agent decision-making control methodology. *IEEE Transactions on Industry Applications*, 2013.

[54] Q. Zhu, L. Bushnell, and T. Başar. Resilient distributed control of multi-agent cyber-physical systems. In *Control of Cyber-Physical Systems.* 2013.