An Architecture for the Development of Ambient Intelligence Systems Managed by Embedded Agents

Carlos Eduardo Pantoja
CEFET/RJ
Universidade Federal Fluminense
e-mail: pantoja@cefet-rj.br

Heder Dorneles Soares and
José Viterbo
Universidade Federal Fluminense
e-mail: hdorneles,viterbo@ic.uff.br

Amal El-Fallah Seghrouchni
Sorbonne Universités
UPMC Univ Paris 06, LIP6
e-mail: amal.elfallah@lip6.fr

***Abstract*—Ubiquitous systems consider the use of electronic components for enhancing daily objects with some kind of computational intelligence for aiding users in their tasks pervasively. Ambient Intelligence (AmI) is a branch of ubiquitous computing that provides an environment full of interconnected devices and it can provide data communication, inference mechanism based on context information and collaboration among system's devices. Similarly, the Internet of Things (IoT) provides uniquely identified devices or things in a network for helping users in their activities. Multi-Agent Systems (MAS) are intelligent systems where agents are responsible for reasoning, competing and using resources to achieve desirable goals pro-actively and autonomously. Agents have been employed in some approaches and works during the last years, but none of them considered embedded MAS responsible for smart devices in an AmI system running over an IoT network. Besides, it is also interesting that agents of the embedded MAS can interact, sharing information with agents situated in another embedded MAS using the IoT network to learn from user's experiences. This paper proposes an architecture for the development of AmI systems using embedded MAS for interfacing with sensors and actuators in a heterogenous network using an IoT middleware.**

I. Introduction

Ubiquitous Computing or pervasive computing is the capability of embedding intelligence in everyday objects in a way that the person who interacts with this object reduces the level of interaction with the device or even does not notice it [1]. The tendency of Ubiquitous Systems, supported by the advances in communication and network interconnection, is to allow everyday objects to interact with humans pervasively and to communicate with each other [2]. There are several fields and applications that will be impacted by the use of the Internet of Things (IoT), such as Ambient Intelligence (AmI).

Some questions about the development of AmI solutions refer to technologies that are necessary in the process of communication between devices, node synchronization, collecting and storing context data and inferences for decision making. In general, the objects that participate in this system are usually sensors or embedded devices, which have several limitations regarding hardware resources and the communication capability [3]. In an AmI system where the number of devices can grow exponentially, the concern with scalability becomes latent. To address these limitations, there is *ContextNet* middleware [4], which treats the communication and connectivity of Mobile Nodes (MN) in a scalable way using data distribution protocols based on the DDS standard of OMG [5].

DOI reference number: 10.18293/SEKE2018-110

Intelligent agents are entities that can be constructed in both hardware and software and they are able of performing actions in certain environments autonomously and pro-actively. A Multi-Agent System (MAS) is composed of intelligent agents capable of communicating and collaborating — or even competing — for using resources in an environment to achieve conflicting or common goals [6]. The use of the MAS approach applied in AmI is justified by the autonomous characteristics of agents and their application in complex systems, both found in AmI [7]. However, some traditional smart objects are mainly data gathers and senders, and the data is stored and processed in servers, compromising the autonomy of these objects and because of the high dependency on centralized technologies to provide communication and reasoning in such kind of system [8]. The idea of decentralized MAS responsible for cognitive intelligence in distributed computing is being discussed instead of using centralized MAS for creating real autonomous smart objects [9].

This paper proposes an architecture for developing intelligent systems integrating devices using embedded MAS as smart objects for IoT to be used in the AmI domain. In this architecture, it is possible to develop embedded MAS for controlling devices composed of sensors and actuators. Besides, some characteristics of these nodes are: the ability to communicate with other devices apart of the technology employed in them; truly autonomous; and resilient from the IoT network. For this, our smart objects use Jason framework [10] adopting a specific kind of agent able of communicating with others devices using internally the *ContextNet*. A case study will be presented in a laboratory with several devices employing MAS, Java and Android applications. The contributions of this work are: an architecture for developing solutions for IoT and AmI using different devices supported by intelligent agents; and an extension of a well-known Agent-Oriented Programming Language (AOPL) for programming intelligent devices.

This work is structured as follows: in section 2 is presented some necessary concepts for the understanding of the proposal of this paper; section 3 presents the proposed architecture, a extension of Jason framework integrating the *ContextNet*; in section 4 it is presented a case study e some experiments; in section 5 we discuss some related works and; finally, the conclusions and references are presented.

II. Background

In this section, it is presented the middleware *ContextNet* for IoT, which is responsible for communicating and gathering data from several nodes in a network. Next, some characteristics of the Jason framework [10] is discussed once it is used to develop agent-oriented nodes in the proposed architecture.

A. The Middleware ContextNet

The *ContexNet* is a service for providing context data in stationary and mobile networks. It counts with a Scalable Data Distribution Layer (SDDL) layer [4], which is used for tracking applications in vehicles, industrial automation and data spreading. This service deals with major questions in data communication such as fault tolerance, network load balancing, support for node disconnection, and security. It also provides creation resources and dynamic management for groups. The *ContexNet* middleware works in a *publish/-subscriber* model. The data transferring occurs by using two protocols: the MR-UDP [11] and the OMG DDS [5]. The MR-UDP treats messages between a client and a gateway; the DDS is responsible for distributing data in the core of the network. By using *ContextNet*, it is possible to enable the growing of a network, ensuring the scalability of the content distribution between millions of devices.

B. Framework Jason

The Jason is a *framework* for developing MAS using a cognitive model named *Belief-Desire-Intention* (BDI) [12] and has an interpreter for the BDI based AOPL *AgentSpeak* in Java language. The BDI allows agents to reason based on perceptions captured of the environment, beliefs that represent the knowledge obtained during their existence and they are able of communicating with each other in order to exchange information or achieve mutual or conflicting goals. Besides, they can have plans composed of actions that are activated depending on beliefs on their belief bases.

Specifically, a Jason's standard agent has a reasoning cycle [10] responsible for the processing of all perceptions and beliefs to generate events which activate plans and actions. It is important to understand the reasoning cycle of a standard agent because several extensions (including a proposal in this paper) modify some characteristics of this reasoning cycle to enhance specific kind of agents with new customized abilities. First, the agent captures the perceptions from a simulated environment where agents can interact with virtual objects that may have information represented as perceptions. The agent verifies its mailbox at the beginning of each cycle for existing messages to be read. It is important to remark that the original distribution of Jason does not have any access to real environments. Afterwards, a function updates the *Belief Base* using the captured perceptions from the environment. For each modification in the *Belief Base* it is generated an event that an agent has to deal with to achieve its goals. Then, an event is selected from a list of generated events and when it is selected, it retrieves all the relevant plans of the agent's plan library. After that, a verification is performed to identify which plans can be executed based on its current beliefs and perceptions and a function selects only one plan to be executed. Finally, an action of the selected plan is executed one at a time.

There is an extension of Jason's agents capable of controlling devices such as sensors and actuators connected to microcontrollers named ARGO [13]. This customized architecture is able of capturing perceptions and send them to agents without any interference of the programmer and agents are able of executing actions using actuators without worrying about what kind of hardware is being employed. The hardware and software layers are uncoupled. The reasoning cycle of an ARGO agent uses the Javino [14] for capturing perceptions. The Javino provides a serial communication between microcontrollers and Jason using a basic protocol to guarantee the correct information exchange between the transmitter and the receiver.

An ARGO agent has the abilities, at runtime, of: selecting the microcontroller which it desires to control; deciding whether or not to block the perceptions coming from sensors releasing processing time for other tasks, for example; filtering undesirable perceptions; and limiting the time interval of perceiving the environment. However, ARGO agents only communicate with agents hosted in its MAS. This implies that if a MAS is embedded into a device, the communication will be limited to this device. Therefore, in this paper we propose a new kind of agent that is able of communicating with other agents (and devices) using Jason and *ContextNet* to be used in an architecture for programming pervasive solutions in AmI.

III. The Overall Architecture For AmI Systems Using MAS

In this section, it is presented our architecture for developing AmI systems using real autonomous and embedded MAS as devices. The architecture considers an open and dynamic environment where devices using the agent approach and others devices can enter or leave anytime. This approach leads us to a decentralized and a collective reasoning since every embedded MAS is considered an autonomous and independent thing capable of communicating and negotiating — or even acting as a group pursuing common goals — with others devices. By independent, it means that a smart object embedded with MAS are able to keep running and reasoning even if communication and interaction technologies stop. In fact, we consider this embedded smart object a MAS as a Thing.

Basically, the AmI system uses the *ContextNet* as IoT middleware where every device should connect as a Mobile Node (MN) to be part of the system and to communicate with others devices. Some of these devices can be MN with sensors and actuators, Android devices or electronic smart objects (tv, refrigerator, etc.). In this approach, we assert that MN can employ embedded MAS for managing sensors and actuators in the AmI systems. This MN is composed by a cognitive layer using Jason framework with agents responsible for interfacing with hardware using ARGO; and agents responsible for communication in the network using an instance of *ContextNet* nodes. The hardware layer is composed by: the platform for embedding the MAS, which could be any tiny computer such as Raspberry Pi; sensors and actuators; and microcontrollers. Figure 1 depicts an overview of the proposed architecture.

In some cases, the use of MAS can bring advantages compared to MN that only work as data repeaters sending information from sensors to a server for discovering context about a situation and from MN which need stimulus from other devices to act upon the environment. Agents are pro-active, autonomous and are capable of reasoning about information from the environment that they are situated. These characteristics allow an improved information or even a previous context discovery before sending it to a server , releasing its processing power, for example. Besides, agents can make decisions and act autonomously without depending of a third part processing (if they have sufficient processing power).

Another important characteristic is that a MAS can be programmed individually as a MN that is able to interact with other devices (including another MAS) using a special kind of agent named **Communicator**, which has an instance of *ContextNet*. There is another type of agent responsible for controlling sensors and actuators that can be used along with the **Communicator** one. Therefore, it can exist four types of agents in a project:

- **Standard**: the standard agent is able to communicate with others agents of its MAS but it is not possible to communicate with agents from a different MAS and it is not able of controlling any kind of hardware. It is the basic unit of a MAS.
- **Argo**: it is a customized architecture of agents capable of controlling microcontrollers independently of its type and the domain applied in the solution. ARGO agents have all the abilities of a standard agent but are not able of communication with agents from a different MAS.
- **Communicator**: this agent is able of communicating with agents from a different MAS or any device using *ContextNet*. It has the same abilities of a standard agent but but it is not able of controlling hardware devices.

The *ContextNet* is a scalable architecture, which guarantees a great number of devices transmitting data at same time. In this approach, we propose the use of MAS developed using Jason to exploit some advantages of using a robust middleware for IoT applications and a well-know framework for agents solutions. Jason already counts with implementations of **Standard** and **ARGO** agents. Then, we propose an extension of Jason framework creating a customized architecture for agent **Communicator** with a *ContextNet* instance embedded into its implementation in order to facilitate the use of such kind of agent. If some device needs to communicate with a MN with a MAS, it should send messages in KQML format and translate the received KQML messages. It is important to remark that Jason uses KQML as communication language. We also propose a mechanism for processing messages based on KQML performatives in the following section.

A. Extending Jason using an IoT Middleware

In order to allow the programming of agents that are able of communicating through IoT, a special kind of agent named **Communicator** was proposed. This agent is responsible for exchanging messages between agents hosted in different MAS or in any other device. For this, all the devices must be connected to the *ContextNet* middleware and the **Communicator** agent must have a communication mechanism for sending and receiving messages through the IoT. Thus, the reasoning cycle of the **Communicator** agent was extended with the *ContextNet* embedded in its architecture (Figure 2).

The first modification happened in the beginning of the reasoning cycle and it is capable of receiving messages from others devices using *ContextNet* and messages coming from agents of its own MAS using the *checkMail* method. All messages received are processed to generate events and update the agent's *Belief Base*. The next modification was inserted at the end of the reasoning cycle after the *sendMsg* step. In this moment, the agent can send a message to agents hosted in its MAS or to another device in IoT using *ContextNet*.

A message can be sent to another **Communicator** agent or any device able of understanding the message format. Every agent must have a unique identification number provided by *ContextNet* and to send any message, the agent uses an internal action named **sendOut** likewise the original internal action **send** from Jason. Both of them send a message to an addressee using a illocutionary force. The major difference between them is that **sendOut** sends a message to a mobile device or a **Communicator** agent in another MAS. In this paper, the available illocutionary forces are:

- **achieve**: sends a goal to be accomplished by the addressee. The content of the message sent will be inserted in the base of intentions of this agent.
- **unachieve**: drops a goal in case it has not been reached yet. The content of the message will be removed from the base of intentions of the addressee.
- **tell**: sends a belief of the sender that the addressee believes to be true. The content of the message must be a literal, which represents a belief and will be inserted into the belief base of the addressee.
- **untell**: the sender agent informs the addressee agent that the belief is no longer to be believed. The content of the message is removed from the belief base of the addressee.

In order to integrate *ContextNet* into Jason architecture, some modifications were performed. First of all, the *Communicator* class for creating an agent with the ability of communicating was added as an agent's customized architecture. This class has an attribute *commBridge*, which is responsible for sending and receiving messages from *ContextNet*. This class also has a function for adding the received message from *ContextNet* to the agent's mail box. The *commBridge* implements a process for mounting and verifying a message to

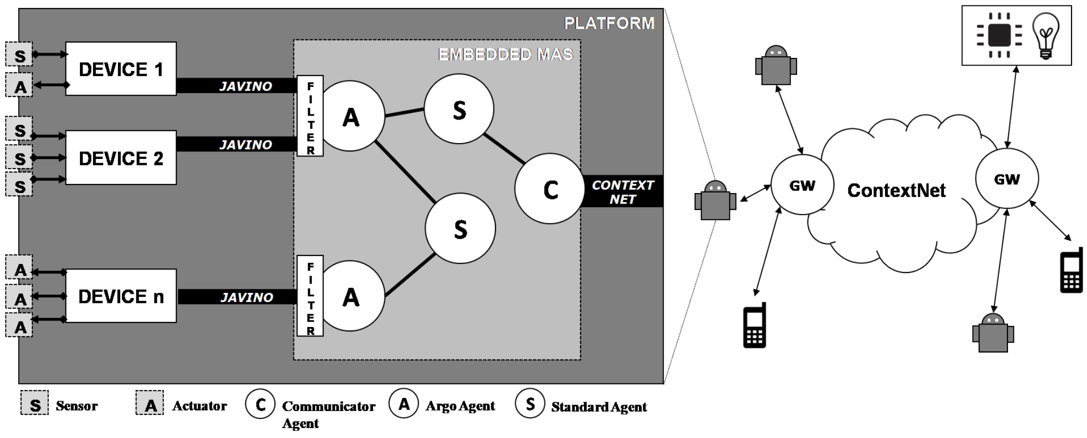

Fig. 1. An overview of the proposed architecture.

guarantee no losses of data in the communication. A message is composed of the following fields: a pre-amble to identify the origin of the message with a length of 4 bytes; fields to identify the sender and the receiver of the message with 32 bytes each; the identification of the illocutionary force with 32 bytes; and the message content with 256 bytes.

When the sender starts sending the message, the size of all fields are calculated to identify the beginning and the end of each field. After that, the pre-amble is added at the beginning of the message to verify the origin of the message. The message is mounted adding all the fields in a single string message that is sent by *ContextNet*. When the receiver receives the message, the pre-amble is verified to guarantee the origin of the message. Then, all the fields' size is verified to guarantee no losses in the communication process. If everything is ok, the message is mounted and processed as a Jason's message. Otherwise, the message is discarded.

The native *TransitionSystem* class of Jason was modified in the reasoning cycle function for allow to check if exist messages to be read coming from *ContextNet*. The existing messages are added to the mail box of the agent to be processed as beliefs or intentions depending on the illocutionary force related to the message as explained before. After that, the agent can send a message using an internal action named *sendOut*, which uses the *commBridge* to send a message using the *ContextNet*. Another internal action named *setMyCommId* is responsible for setting the identification string used by *ContextNet* to identify uniquely a device in the IoT. It is important to remark that all modifications proposed do not interfere in the original Jason distribution nor in ARGO.

IV. Case Study

In this section, we present an initial study case based on the proposed architecture using the Jason and the *ContextNet* middleware in order to control some functionalities in a laboratory. The following scenario explicit the general behavior of our approach: Kate is the head of the laboratory and she wants to get information in her smartphone about some features such as the temperature, luminosity and the status of the lights (on or off). She also wants to know if there are students using the laboratory while she is away receiving an updated list every time she requests. Only students which have an authorization can enter in the laboratory. So, the student's will request access to the laboratory and use its smartphone. Besides, every student when entering the laboratory should connect to the network to inform of his or her presence.

The laboratory is equipped with some devices managed by MAS and MN using Java language and Android. The temperature and luminosity sensors (LM35 and LDR respectively) are controlled by a MAS using a Galileo Intel Gen 2. For controlling status and activation of lights, a MAS running in Raspberry Pi Zero and an Arduino UNO assembled with an ATMEGA328 microcontroller were employed. The electrical installation of the laboratory was modified to accept commands coming from the microcontroller. The door is controlled by a MAS embedded into a Raspberry Pi and is responsible for registering the students entrance and to verify if they have the permissions to enter the laboratory accessing a remote database. The Android applications were developed implementing Android MN from *ContextNet*.

Every device has only one MAS embedded in a tiny computer such as Raspberry and Galileo since they have enough processing power to host embedded MAS. ARGO agents are responsible for controlling actuators and sensors that are connected to one or more microcontrollers. In our case, we assembled an Arduino UNO board, an ATMEGA microcontroller, and the Galileo GEN's GPIO. The **ARGO** agents send serial messages to the microcontroller to some action to be performed in the laboratory. On the other hand, every MAS can employ **Communicator** agents. It is important to remark that every MAS should have only one **Communicator** agent, because it will be responsible for the identification in the IoT network of the device; to communicate with other agents; and to send and receive data from *ContextNet* gateway.

The MAS responsible for controlling the temperature and luminosity sensors are composed of one **ARGO** agent and one **Communicator** agent. The former one is responsible

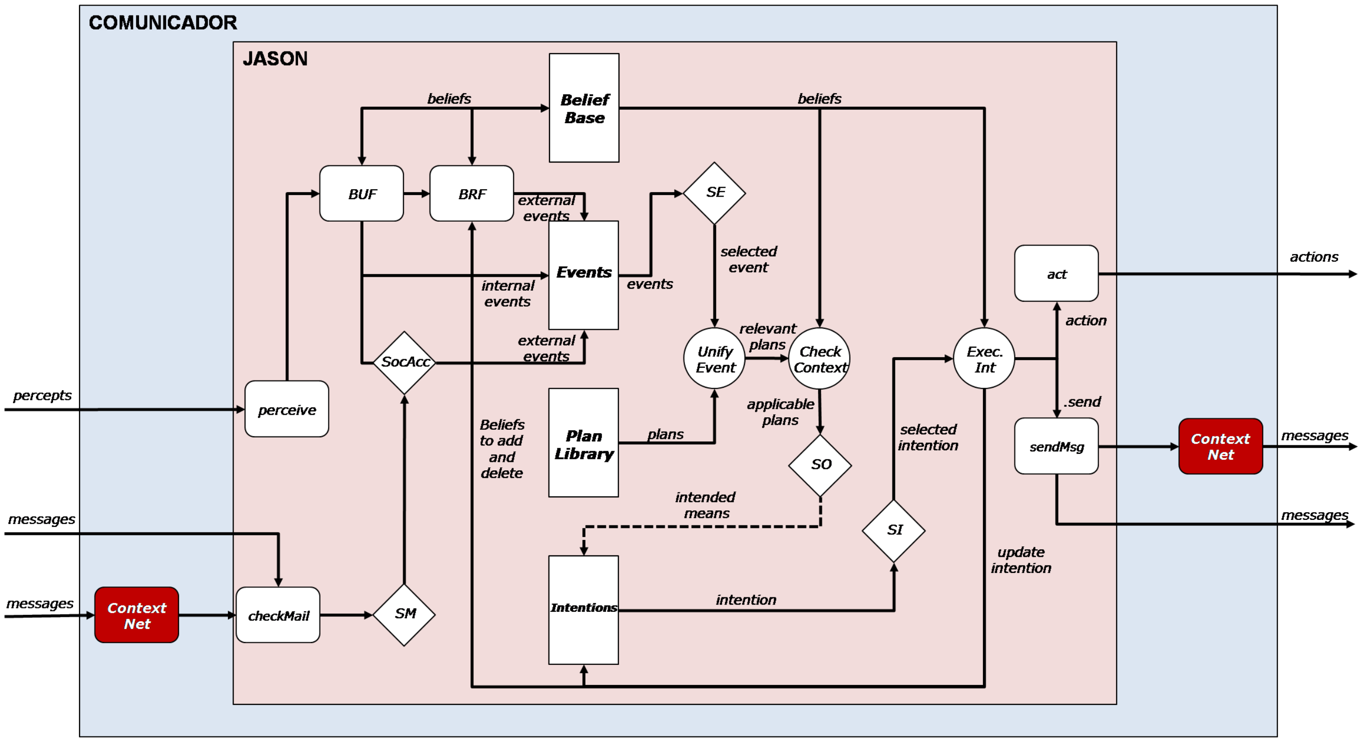

Fig. 2. The Reasoning Cycle of a Communicator agent.

for capturing the perceptions from sensors and to transmit them to the later one, which manages the received requisitions from other smart devices and respond to them with the most current perceptions using *ContextNet*. The MAS responsible for controlling the door is composed of three agents: **ARGO**, **Communicator**, and **Standard**. The **ARGO** agent is able of opening the door if it is closed and for retrieving the status of the door (opened or closed) when requested. The **Communicator** receives the requisitions from other devices and it sends to the **Standard** agent the requisitions that need to be verified if the user has access permission using a customized internal action for accessing the database. If the user has the access permission, the agent sends a message to the **ARGO** agent for opening the door.

The MAS that controls the activation of lights and other electrical stuff has an **ARGO** and a **Communicator** agent. The former one interfaces with the hardware that controls the electrical installation, and it is capable of retrieving perceptions about the status of the lights or it receives commands for activating resources (light, air-conditioning, etc.). The **Communicator** just manages requests coming from other devices like the previous examples. The Android application for the head of the laboratory employs functionalities for listing every access registered in the laboratory; status of electrical stuff and door; commands for activation and deactivation; and some administrator function such as registering a new user and setting new permissions. The user's Android application communicates with the MAS of the luminosity and temperature sensors in order to get these information and to define the preference for the temperature (warm, hot or cold); and it asks for accessing the laboratory. In fact, every Android application uses an instance of *ContextNet* for MN.

V. Related Work

It is possible to find several works that try to integrate MAS in AmI systems. However, these solutions only provide communication with agents originally from its MAS, avoiding communication with other agents or embedded MAS that can enter in the AmI system eventually The Agent of Things (AoT) [15] is a definition for devices or things that are managed by a single agent in dynamic environments. The authors suggest that in such kind of environments, the communication between devices are highly programmable and depends on a previous interaction configuration. It is proposed a conceptual framework which consider a direct physical interaction between devices using the hardware layer and using a software layer where every agent represents a device. However, the framework is conceptual and agents are centralized in a software layer. In our approach, we consider a MAS as a thing and a real laboratory implementation as proof-of-concept.

In [16], it is discussed the use of MAS in IoT rising questions about communication and the use of protocols where the main objective is to implement functionalities for access control in a agent-based IoT. The architecture uses a central server for controlling and coordinating agents and it is also presented a hybrid system containing intelligent agents and IoT devices in traffic scenario. There is one embedded agent for each device and the authors do not explicit how agents are organized. The Agent-based Cooperating SO (ACOSO) is used as IoT middleware for providing an IoT network where agents are devices and the whole group of agents is a MAS [17]. The ACOSO supports the development of MAS in the level of things. So, every smart object can be abstracted to a cooperating agent using Jade as AOPL. The agents are able of managing sensors and actuators; reasoning and decision making using local and distributed databases; and a communication system for the interaction between smart objects. However, the environment is closed and it is not possible of new devices to enter in the system and there is only one agent per device.

A decentralized approach with a framework and an architecture for engineering IoT applications based on autonomous smart objects is proposed in [8]. The authors defend that traditional smart objects are data gathers and senders and the data is stored and processed in central servers. The proposed smart objects are capable of running even if remote technologies (i.e. gateways) are not available. There is one agent for smart object and it is programmed using XML technologies (it does not use AOPL) in a web platform named Eve agent.

VI. Conclusion

This work presented an architecture for the development of AmI systems employing the agent approach and supported by an IoT middleware named *ContexNet*. In this architecture, it is possible to assemble devices, which have embedded MAS responsible for controlling sensors and actuators and for communicating with other devices. Every device is an independent solution and it is free to enter and leave in the architecture. The proposed approach uses Jason framework for the development of the MAS. Besides, we enhanced a real laboratory with such technologies to use it as a proof-of-concept application. This architecture aims to provide a kind of framework for the development of AmI systems.

By using the proposed architecture, it is possible to create an AmI system, which allows communication among several devices at real time, where all devices connected can change information, offering flexibility in the choice of which technology employed in the solution. From MN — which do not use agents — the data information is transmitted to the *ContextNet* gateway in raw format. So, the discovery of context situations coming from these nodes only will happen exclusively in the server. In some cases, it could be necessary that the context and the reasoning for specific situations happen directly in the devices. In cases where the device is embedded with a MAS, a previous reasoning can be performed using the raw information captured as perceptions from **ARGO** agents reducing the processing in the server and avoiding bottlenecks.

This work also presented an extension of Jason to allow embedded MAS in mobile devices to communicate with other agents hosted and embedded in different devices. It provides a specific and new kind of agent that has the ability to communicate with other agents using *ContextNet*. In this type of agents, the middleware is part of the reasoning cycle of the agent, which uses internal actions for sending messages to agents with the same ability and hosted in a different MAS. For future works, the architecture will be employed in other laboratories containing different devices and technologies. We also aim to improve and formalize the ability of infer rules in the *ContextNet* middleware. Besides, we also aim to create a *testbed* for automation of experiments and resource sharing, in order to assist in the validation of new proposals involving IoT technologies and MAS.

References

[1] M. Weiser, “The computer for the 21st century,” *IEEE pervasive computing*, vol. 1, no. 1, pp. 19–25, 2002.

[2] D. Surie, O. Laguionie, and T. Pederson, “Wireless sensor networking of everyday objects in a smart home environment,” in *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on.* IEEE, 2008, pp. 189–194.

[3] H. D. Soares, R. P. de Oliveira Guerra, and C. V. N. de Albuquerque, “Ftsp+: A mac timestamp independent flooding time synchronization protocol,” in *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC.* Sociedade Brasileira de Computação, 2016, pp. 820–832.

[4] M. Endler, G. Baptista, L. Silva, R. Vasconcelos, M. Malcher, V. Pantoja, V. Pinheiro, and J. Viterbo, “Contextnet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking,” in *Proceedings of the Workshop on Posters and Demos Track.* ACM, 2011, p. 2.

[5] G. Pardo-Castellote, “Omg data-distribution service: Architectural overview,” in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on.* IEEE, 2003, pp. 200–206.

[6] M. Wooldridge, *An Introduction to MultiAgent Systems.* Wiley, 2009.

[7] C. Maciel, P. C. de Souza, J. Viterbo, F. F. Mendes, and A. El Fallah Seghrouchni, *A Multi-agent Architecture to Support Ubiquitous Applications in Smart Environments.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 106–116.

[8] M. E. P. Hernández and S. Reiff-Marganiec, “Towards a software framework for the autonomous internet of things,” in *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on.* IEEE, 2016, pp. 220–227.

[9] M. P. Singh and A. K. Chopra, “The internet of things and multiagent systems: Decentralized intelligence in distributed computing,” in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on.* IEEE, 2017, pp. 1738–1747.

[10] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason.* John Wiley & Sons Ltd, 2007.

[11] L. Silva, M. Endler, and M. Roriz, “Mr-udp: Yet another reliable user datagram protocol, now for mobile nodes,” *Monografias em Ciência da Computação, nr*, vol. 1200, pp. 06–13, 2013.

[12] M. E. Bratman, *Intention, Plans and Practical Reasoning.* Cambridge Press, 1987.

[13] C. E. Pantoja, M. F. Stabile, N. M. Lazarin, and J. S. Sichman, “Argo: An extended jason architecture that facilitates embedded robotic agents programming,” in *Engineering Multi-Agent Systems: 4th International Workshop, EMAS 2016*, M. Baldoni, J. P. Müller, I. Nunes, and R. Zalila-Wenkstern, Eds. Springer, 2016, pp. 136–155.

[14] N. M. Lazarin and C. E. Pantoja, “A robotic-agent platform for embedding software agents using raspberry pi and arduino boards,” in 9^{th} *Software Agents, Environments and Applications School*, 2015.

[15] A. M. Mzahm, M. S. Ahmad, and A. Tang, “Enhancing the internet of things (iot) via the concept of agent of things (aot),” *Journal of Network and Innovative Computing*, vol. 2, no. 2014, pp. 101–110, 2014.

[16] D. Rivera, L. Cruz-Piris, G. Lopez-Civera, E. de la Hoz, and I. Marsa-Maestre, “Applying an unified access control for iot-based intelligent agent systems,” in *Service-Oriented Computing and Applications (SOCA), 2015 IEEE 8th International Conference on.* IEEE, 2015, pp. 247–251.

[17] C. Savaglio, G. Fortino, and M. Zhou, “Towards interoperable, cognitive and autonomic iot systems: an agent-based approach,” in *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on.* IEEE, 2016, pp. 58–63.