IoT-a, embedded agents for smart Internet of Things Application on a Display Wall

Florent Carlier and Valérie Renault
Université du Maine,
Avenue Olivier Messiaen,
72085 LE MANS cedex 9, France
Email: firstname.name@univ-lemans.fr

***Abstract*—Our goal is to bring smart algorithms as close as possible to the hardware level of physical objects. We propose to consider smart objects as "Internet of Things - Agents" (IoT-a) in which each object follows, in its design and in its core, multiagent protocols and architecture. We focus our work on an embedded low-level multiagent architecture - named Triskell3S - and its underlying communication protocols. A key challenge for the IoT is to allow the components to interact together, which requires common protocols of communication and interoperability. Our proposal is to study the interaction in the oriented agent approach, generally used in multiagent based simulations. In order to experiment the Triskell3S architecture in a real context, we propose a reconfigurable display wall where each screen module is composed by a screen and an embedded card. Each IoT-a is then a smart screen module that communicates with the other wall modules to display a synchronized video. Finally, our benchmark to test the efficiency of our communication protocols relies on the distributed approach of N-puzzle solving. The agent interactions on the display wall allow to recompose a video based on all the premixed screens in real time.**

I. Introduction

By browsing the literature, we can notice that the concept of IoT hides and covers different meanings, contexts of applications and, consequently, various architectures. Many challenging issues have to be addressed behind this term. Many researches were made on IoT [1] in particular works about the control of the smart objects [2], its reasonning, sensors, effectors and resources states [3]. Since years [4], it is well known that many concepts and issues from researches on multiagent systems were found in the area of IoT (for example, in terms of interactions, communication protocols, interoperability or behaviors). Our main problematic is how to develop some middleware based solution to allow the objects to interact to each other. In our article, we envisage the *Things*, the objects, as the *Agents* of multiagent systems in order to model and implement a multiagent architecture for IoT. We propose to consider smart objects as "Internet of Things - agent" (IoT-a). Each object follows, in its design and in its core, multiagent protocols and architecture [5]. This article is organized as follows. In Section 2, we present and compare the different IoT paradigms and the links made between the multiagent systems and the IoT. Section 3 focuses on our Triskell3S multiagent architecture. This platform allows us to define IoT-a as defined in the MQTT and D-Bus protocols and the FIPA-ACL standards. Next, in Section 4, in order to experiment the Triskell3S architecture in a real context, we present a reconfigurable display wall relying on IoT-a screen modules. Finally, through the example of distributed N-puzzle solving, we show that the interaction-oriented approach of the agent, initially based on simulation, is adapted to the IoT-a.

II. Multiagent based simulation and the Internet of Things context

A large subset of multiagent platforms have already implemented some plugin for mobile and fixed environment or specific plugin for the domain of the IoT or *ambient intelligence* context. Most of the time, the middleware is developed to be between the harware and the multiagent application level. This multiagent level is a simulation based system, the core of the platform is then not adjusted to the embedded constraints.

In the smart objects area, the middleware architectures are gaining more and more importance for the IoT with the adoption of the Service Oriented Architecture (SOA) principles **[Fig. 1]** [6]. Five main layers compose this middleware : at the top of the architecture, the application layer is dedicated to a given problem; at the bottom, physical objects are in interaction with their environment. Between these two layers, the Service Composition, the Service Management and the Object Abstraction, enable to provide specific services from the object abstraction to the final user.

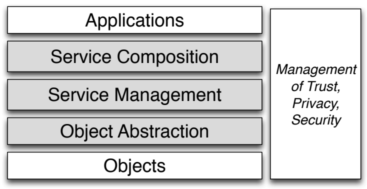


Fig. 1. SOA-based architecture for the IoT middleware

Multiagent platforms can be seen as the application layer of the SOA. Generaly, this layer is not embedded in a real physical component. Many researches have be done in the context of ambient intelligence, in particular, in the context of smart home environments or smart grids [7]. However, in many of these researches, the intelligence is delocalised and processed in a centralized server. The environment is composed of captors and effector without real intelligence and interactions between them. As in [8], the goal of our research is to encapsulate the multiagent architecture into the physical components.

III. Triskell3S : Embedded architecture for interaction oriented agents

Triskell3S is an embedded architecture for interaction-oriented agents [5]. This architecture is consistent with the vision of IoT defined by the European Commission in [1] : ”Things having identities and virtual personnalities operating in smart spaces using intelligent interfaces to connect and communicates within social, environmental, and user contexts.”

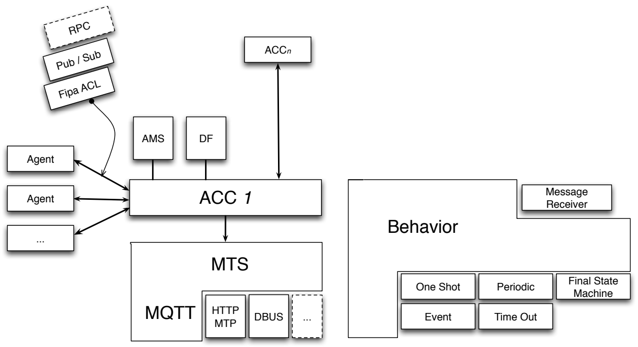


Fig. 2. The embedded multiagent architecture of Triskell3S

Figure **[Fig. 2]** resumes the embedded multiagent architecture that we propose for our IoT-a. Triskell3S relies on two interaction levels inter-agents and intra-agents. The first level deals with how to implement the MQTT protocol [9] according to the FIPA-ACL communication specifications [10] to allow interactions between the IoT-a levels. In a second level, to control the hardware layer, the inter-process communication within the agent itself is based on the D-Bus protocol [11]. Thereby, we propose to integrate the middleware platform in the agent itself. The interaction with the technological layer relies on a specific hardware protocol, the D-Bus protocol.

IV. Interaction-oriented agents for a reconfigurable display wall

To illustrate our architecture in a real context, we propose to produce a reconfigurable display wall. This wall can interact with mobile systems and interactive devices. Our assumption is that it is possible to build a display wall with a modular architecture. Each module, an IoT-a, is composed of a screen and an embedded card to be autonomous.

The specificity of our research project, named Tifaifai, is to create a smart screen wall which is able to adapt and configure itself according to the usage context of learning and training. The decentralized approach of the Tifaifai project, based on the Triskell3S plateform, allows us to gain in flexibility and scalability at lower cost. As shown in the figure **[Fig. 3]**, for the visualization surface to be adaptive, each screen is associated with an embedded agent controlling the display at any time. For example, a *MonitoringAgent* can interact with a *DisplayAgent* to send a `PAUSE` with the MQTT command. The *DisplayAgent* can propagate the instruction to the *ScreenAgent* via D-Bus so that the video turns on ”pause”.

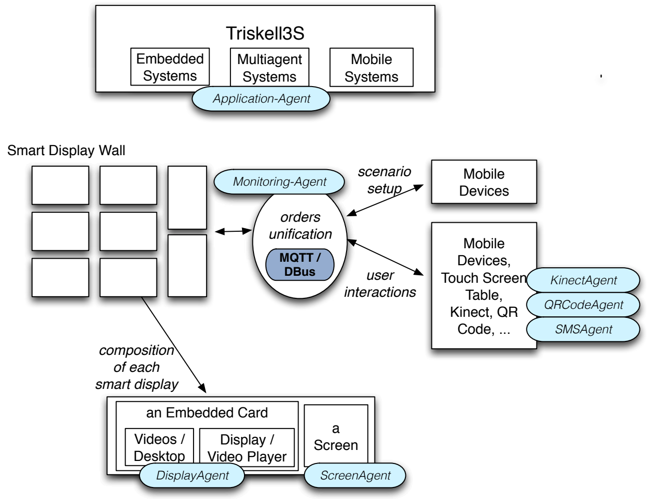


Fig. 3. Implementation of Triskell3S in an Smart Display Wall

If we want the system to be adaptive in real-time, each screen has to be able to ’reflect’ on what it can or has to display, communicate and exchange messages with the other screens. To this end, each screen is linked to an embedded card like a Raspberry Pi card [12]. These cards are gathered into a cluster allowing to either control each screen independantly. This simply means using an embedded card for an application on a single screen, or synchronizing the display between screens, leading to the visualization of a multi-windows application or of a video (for instance) dispatched on N screens. Raspberry Pi cards have already been used to build video walls as in the PiWall software [13]. In this package the wall is a master / slave architecture composed of one card per screen to drive the relevant portion of the wall and a master card to rule them all. The IoT-a model we propose bypasses the mastercard to improve the adaptivity of the system. Each autonomous card controlling an independant screen, the overall coherence between the screens of the wall relies on the communication and interaction between all agent-embedded cards. The screenwall is currently composed of 16 screens, all having a given knowledge regarding their environment and their position relative to each other. This distributed architecture allows hot-plugging/unplugging other screens or embedded systems. The figure **[Fig. 5]** shows a 16 screens set-up presenting videos. Making screenwalls interactive is challenging but necessary if it is to be used as a support to collaborative applications [14]. Some specific agents have been introduce to interact with the video on the wall as the *KinectAgent*, the *QRCodeAgent* and the*SMSAgent*. These interactive agents illustrate the heterogeneity of our IoT-a architecture. These three agents, as their names suggest, allow the user to send commands to the wall, with various devices, as for example : `STOP` (to stop one or several videos), `MOVE` (to move the video1 on an other screen), `PAUSE`, `MUTE_VIDEO`, etc. If the architecture allows this kind of control, the way to interact with the data should be studied according to the user domain and context.

V. Experiments

We experiment our architecture on a distributed approach for N-puzzle solving with the Eco-Problem-Solving Model [15] applied to the display wall. The traditional N-puzzle problem includes a square board containing N square tiles and an empty position named the "blank". The goal is to move the tiles from some random initial configurations into a final one which rebuilds an image. In this problem, each agent is characterized by a state, a goal, a satisfaction behavior and a fight behavior. In our case, to experiment Triskell3S IoT-architecture, the N-puzzle problem is illustrated on the display wall in order to recompose a video. The purpose of this article is to experiment our embedded agent architecture on a traditional distributed problem, in particular to test the efficiency of our communication protocols. In the distributed approach of N-puzzle [15], the global goal G_{puzzle} is decomposed in *n* smallest independant subgoals that must be reached by G_{puzzle} to become satisfied. The subgoals are then to position each tile on their final patch :

$$G_{puzzle} = \{position(\tau_1, p_1), ..., position(\tau_n, p_n)\} \tag{1}$$

where *n* is the number of tiles, the τ_i represents the tiles and p_i their goal patches.

On a video N-puzzle embedded to a display wall, a tile is then represented by a video and the patch by a physical screen. It is necessary to take into account the display software which controls the video on the screen. We then state :

$$G_{puzzle} = \{position(v_1, s_1, d_1),\ ...,\ position(v_i, s_n, d_j)\} \tag{2}$$

where v_i, s_n, d_j respectively represent the video on each tile, the physical screen and the display (the sofware that plays the video). The various indexes i, n and j allow us to have different displays and therefore different videos on one screen. One screen could then support several tiles on its own. The distinction between physical screens and rendering displays is important in the context of embedded systems and in our architecture as it directly affects the management of messages between agents. If we consider this goal in a more general framework of IoT-a, the video is 'just' a data used by the agent, the screen is the hardware which handles the agent and the display manages the communication links between the data and the hardware. We can then describe the subgoal of an IoT-a environnement in the following manner :

$$G_{env} = \{handler(d_1, h_1, c_1),\ ...,\ handler(d_i, h_n, c_j)\} \tag{3}$$

where d_i, h_n, c_j respectively represent the data to be processed, the hardware of the embedded system and the software communication links. As in (2), the indexes could be different.

As in [15], we can change our point of view and now consider the agent a_i, its states - i.e. state(a_i) - and its behaviors - i.e. behavior(a_i) -, as a set of actions intended to make it reach its goal from its current state. This current state is defined by the authors as :

$$\forall g_i \forall a_i / goal(a_i) = g_i, satisfied(g_i) \Leftrightarrow state(a_i) = goal(a_i) \tag{4}$$

Then, in [15], the authors apply this definition to the N-puzzle and describe the tiles τ_i as :

$$\tau_i = \{p_i, p_k, behavior(\tau_i)\} \tag{5}$$

$$satisfied(position(\tau_i, p_i)) \Leftrightarrow p_k = p_i \tag{6}$$

where p_i is the patch goal of τ_i, p_k is the current patch, and behavior(τ_i) a set of actions allowing the tile to slide from its current patch to its goal patch. With (5) and (6), the authors show that the satisfaction of the subgoal relies on the capacities of its tile agent to "do the right thing" for reaching its goal and to propose an agent-oriented description of the problem. However, in the context of the IoT-a, we show in (3) the importance of the communication and interaction. This allows us to shift from an agent-oriented description to an interaction-oriented agent, as defined by the Interaction-Oriented Design of Agent simulations (IODA) formal method [16].

The IODA model consists of abstracting from the agents, the actions they participate in, by reifying them into the notion of interaction. The authors propose to define an interaction matrix between source and target agents. Each cell of the matrix corresponds to the interactions that a source agent can perform on a set of target agents.

In our context, the IODA model, initialy based on simulation, is extended to an embedded environment. We propose to adapt the IODA matrix to two interactions levels. First, the definition of the interaction inter-agents represents the communication between all the IoT-a. In terms of implementation, this level relies on the MQTT communication protocol. Then, we define the interaction intra-agents (a smart module) allowing the communication between the hardware and the software. The inter-process communication within the agent itself is based on the D-Bus protocol. We can see that this matrix is also compatible with the SOA organisational levels defined in figure **[Fig. 1]**. Figure **[Fig. 4]** shows an adaptation of the matrix of interactions in the context of IoT-a applied to the resolution of N-puzzle problem on the display wall.

In order to verify the efficiency of our architecture and communication protocols, we experimented the N-puzzles problem on the display wall of 4*4 screen modules and of 3*3 screens, as illustrated in **[Fig. 5]** .

The average solving solution lengths and number of moves made by each tile time is equivalent to what is found in the literature of Eco-Problem Solving [15]. Once the resolution

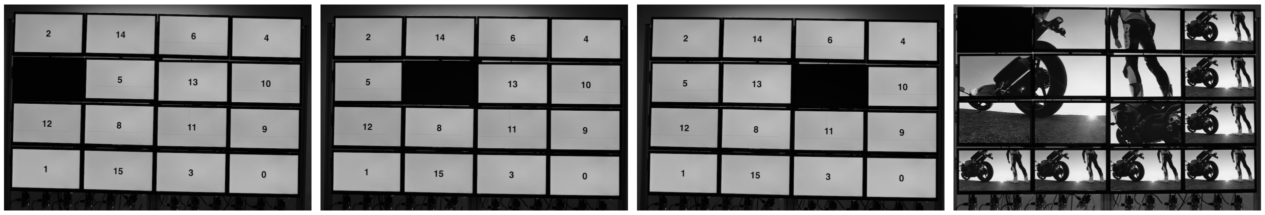
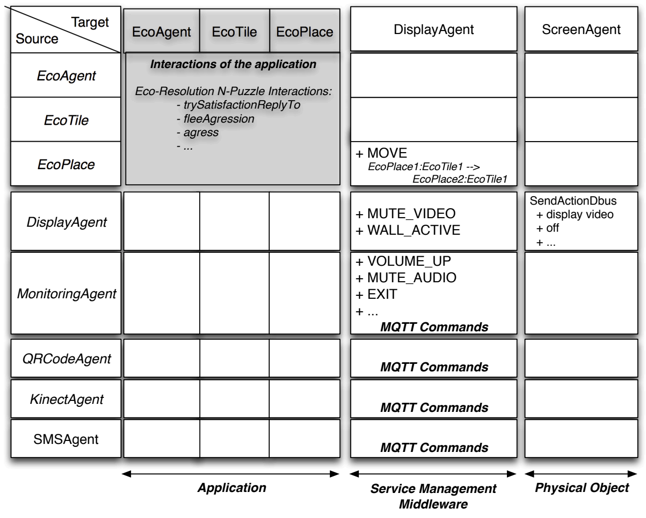


Fig. 5. Distributed resolution of N-Puzzle on the smart display wall. On the first 3 pictures, each video displays the number of a tile, their goals are to present the numbers in the right order. On the last picture, the N-puzzle problem is on the 3*3 display wall with a video. The screens on the right side and bottom present the reference video.



Source \ Target	EcoAgent	EcoTile	EcoPlace	DisplayAgent	ScreenAgent
EcoAgent	***Interactions of the application*** *Eco-Resolution N-Puzzle Interactions:* - *trySatisfactionReplyTo* - *fleeAgression* - *agress* - ...				
EcoTile					
EcoPlace				+ MOVE *EcoPlace1:EcoTile1 --> EcoPlace2:EcoTile1*	
DisplayAgent				+ MUTE_VIDEO + WALL_ACTIVE	SendActionDbus + display video + off + ...
MonitoringAgent				+ VOLUME_UP + MUTE_AUDIO + EXIT + ... ***MQTT Commands***	
QRCodeAgent				***MQTT Commands***	
KinectAgent				***MQTT Commands***	
SMSAgent				***MQTT Commands***	
	Application			***Service Management Middleware***	***Physical Object***

Fig. 4. IODA matrix of interactions in the context of a-IoT. Example with the resolution of N-puzzle problem on the display wall

is finished, all the screens stay synchronised to compose the global video.

VI. Discussion and Future Work

The main contribution of this article is to implement an interaction oriented agent approach in a real environment, a smart display wall. By this way, we show that the multiagent based simulation models can be extended to the Internet Of Things context, via what we named an IoT-a, for ”Agent - Internet of Things”. This approach, with the communication protocols MQTT and D-Bus, allows us to bring the agents as close as possible to the hardware component. The next step is to experiment this architecture with the user interactions on the screen wall. Researches on the way to interact and navigate in large information space on remote displays have shown that new types of interaction have to be found.

References

[1] I. D. N. Enterprise, R. I. G. Micro, and N. systems, *Internet of Things in 2020, roadmap for the future, version 1.1*, co-operation with the working group RFID of the ETP EPOSS, Ed., 2008.

[2] F. Kirchbuchner, T. Grosse-Puppendahl, M. R. Hastall, M. Distler, and A. Kuijper, “Ambient intelligence from senior citizens' perspectives: Understanding privacy concerns, technology acceptance, and expectations,” in *Ambient Intelligence - 12th European Conference, AmI 2015, Athens, Greece, November 11-13, 2015, Proceedings*, 2015, pp. 48–59. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26005-1_4

[3] S. Mamidi, Y.-H. Chang, and R. Maheswaran, “Improving building energy efficiency with a network of sensing, learning and prediction agents,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 45–52. [Online]. Available: http://dl.acm.org/citation.cfm?id=2343576.2343582

[4] F. Piette, C. Caval, A. E. Fallah-Seghrouchni, P. Taillibert, and C. Dinont, “A multi-agent system for resource privacy: Deployment of ambient applications in smart environments (extended abstract),” in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, 2016, pp. 1445–1446. [Online]. Available: http://dl.acm.org/citation.cfm?id=2937202

[5] V. Renault and F. Carlier, “Triskell3s, une plateforme embarque multi-agents pour les iot-a,” in *Journes Francophones sur les Systmes Multi-Agents (JFSMA'2016) - forthcoming*, 2016.

[6] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey.” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/cn/cn54.html

[7] N.-Y. Chong and F. Mastrogiovanni, *Handbook of Research on Ambient Intelligence and Smart Environnements : Trends and Perspectives*. IGI Global, 2012.

[8] J. Jamont, L. Médini, and M. Mrissa, “A web-based agent-oriented approach to address heterogeneity in cooperative embedded systems,” in *Trends in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection, 12th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2014) Special Sessions, Salamanca, Spain, June 4-6, 2014.*, 2014, pp. 45–52.

[9] V. Lampkin, W. T. Leong, L. Olivera, N. Subrahmanyam, and R. Xiang, *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. IBM WebSphere, September 2012.

[10] F. consortium, *FIPA Communicative Act Library, Specification and FIPA ACL Message Structure Specification*. Technical report, 2003.

[11] *The D-Bus home page*. https://www.freedesktop.org/wiki/Software/dbus, 2016.

[12] E. Upton and G. Halfacree, *Raspberry Pi, User Guide*. John Wiley & sons, 2012.

[13] A. Goodyear, C. Hogben, and A. Stephen, *PiWall, An innovative video wall system*, 2015. [Online]. Available: http://www.piwall.co.uk/

[14] O. Chapuis, A. Bezerianos, and S. Frantzeskakis, “Smarties: An input system for wall display development,” in *Proceedings of the 32nd international conference on Human factors in computing systems*, ser. CHI '14. ACM, 2014, pp. 2763–2772. [Online]. Available: http://doi.acm.org/10.1145/2556288.2556956

[15] A. Drogoul and C. Dubreuil, “A distributed approach to n-puzzle solving,” in *Proceedings of the Distributed Artificial Intelligence Workshop*, Seattle, United-States, 1993.

[16] Y. Kubera, P. Mathieu, and S. Picault, “Interaction-oriented agent simulations : From theory to implementation,” in *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*. Patras, Greece: Ghallab, Malik and Spyropoulos, Constantine and Fakotakis, Nikos and Avouris, Nikos. IOS Press, July 2008, pp. 383–387.