IJCSI International Journal of Computer Science Issues, Volume 17, Issue 6, November 2020
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.5281/zenodo.4431052

Agent-based module for simulating mutual exclusion algorithms

SENOUA César [1], KAMLA Vivient Corneille[2], YENKE Blaise Omer[3], KAMGANG Jean-Claude[4]

[1]National institute of cartography, P.O. Box 157 Yaoundé, Cameroon

[2]Dep. of Mathmatics and Computer Science, ENSAI, University of Ngaoundere, Cameroon

[3]Dep. of Computer Science, University of Ngaoundere, Cameroon

[4]Dep. of Mathmatics and Computer Science, ENSAI, University of Ngaoundere, Cameroon

Abstract

In this paper, we were able to model and simulate mutual exclusion algorithms using an SMA approach. Mutual exclusion is one of the fundamental paradigms of distributed systems to ensure consistent access to resources shared between several entities in the system. It ensures that not more than one entity can run a shared resource, called the critical section (security property) and that any request for access to the critical section will be satisfied within a finite time (liveliness property). We have represented an agent as an entity of the system in the modeling. The JADE platform is used for simulation mutual exclusion algorithms. Communication between agent's present limits for resource management critical by a large number of entities using the agent directory service. We were able to optimise the communication platform between our agents, defining message formats and proposing efficient data structures. Analysis of the results of the simulation shows the properties of the mutual exclusion algorithms are satisfactory.

Keywords: *Simulation, Distributed Algorithm, JADE, Mutual Exclusion.*

1. Introduction

Multi-agent systems (MAS) are one of the areas increasingly used in computer research as a system for solving complex problems. This is especially true for the field of simulation [5]. Agent-based simulation offers the possibility to directly represent the simulated entities, their behaviors and their interactions [9]. Indeed, the computing concept of MultiAgent Systems is very well adapted to perform simulations of systems composed of entities [11]. An example of mutual exclusion is the situation of a system in which N processes sharing a physical (a printer for example) or logical (a file) resource. In order to avoid inconsistent situations, the shared resource can only be used by one process at a time, in other words, the resource must be used on a mutually exclusive (MS) basis. Several works on mutual exclusion algorithms exist in the literature, but due to the constraints of large-scale environments, many of the algorithms in the literature are unsuitable, or at least difficult to implement [12]. Thus, the analysis of research results in the field of study of mutual exclusion algorithms is done by simulation.

The objective of this work is to propose an agent module from JADE to model and simulate the mutual exclusion algorithm. JADE has already almost all the elements for the realisation of distributed agent-based applications. The only problem is that it has no examples of implementation of classical mutual exclusion algorithms and of grouping in the literature for users. The interest of this module is multiple. First, this module serves as an introduction to the operation of the JADE API for users new to JADE, in order to see how to realise an implementation of distributed mutual exclusion algorithms. Second, it allows scientists using JADE to test their classical or group distributed mutual exclusion algorithms. Third, it also aims at optimising the infrastructure. As in reference [7] JADE does not adapt properly due to the limitations of the message transport and the agent directory service. We defined the data structures during the writing of the algorithm implementations to facilitate the efficient execution of agents to overcome performance bottlenecks.

2. Related works

There are a large number of distributed simulation tools: there are both commercial products and software in the public domain. It should be noted that this list is not exhaustive, and that there are also other tools that have been used very successfully to build various applications.

The SimGrid[1] project [10] allows the study of scheduling algorithms on heterogeneous platforms. It is a tool that provides basic functionality for the simulation of heterogeneous distributed applications in distributed environments. The specific objective of the project is to facilitate research in the field of programming distributed and parallel applications on distributed computing platforms from a simple network ranging from workstations to computing grids.

Sinalgo [2] is a simulation platform for testing and validating distributed algorithms. This platform is written in JAVA. It is initially dedicated to protocols for wireless sensor networks.

Network Simulator [1] is a computer network simulation software tool, it is one of the most widely used simulators in research laboratories, to simulate and study the performance of network protocols. It provides a platform for developing new protocols and testing them.

The DAJ [3] development tool is used to implement, test, simulate and visualise distributed algorithms in java.

We note that simulators are tools that are useful in the study of distributed algorithms. They allow experimenters to test or validate their algorithms under experimental conditions. However, none of these tools are standardized and are object oriented, yet there are multi-agent concepts that are of increasing interest in the field of simulation.

3. Problem description

In the field of simulation, researchers are often confronted with difficulties related to tools. When they want to simulate their distributed mutual exclusion algorithms on a simulation platform in order to analyse the results of the execution, they face some difficulties. Firstly, having access to certain experimental platforms (for example Grid'5000) is not always obvious and also difficulty in reproducing the results. Secondly, proprietary platforms (e.g. Parsec does not make its code or documentation available) do not allow users to better understand the low-level mechanisms used by the environment.

In addition to all these problems and proposed solutions, the other major problem is the ignorance of the existence of certain simulation platforms that are not yet very well known. And in fact, when these platforms are not available or rare to find it slows down the work. In the end, some people end up giving up on developing their algorithms on simulation platforms.

Faced with these problems, we propose to exploit simulation platforms that provide functionalities, independent of a specific application, allowing users to make modifications to meet more specific needs. It is in this context that we have studied the multi-agent simulation platform JADE, which provides a toolbox, to propose an agent-based module for the simulation of distributed mutual exclusion algorithms.

4. A package for the simulation of mutual exclusion distributed algorithms

4.1 Approach

JADE offers an agent structure to build a multi-agent system, but managing access to a resource shared between several agents is not easy. To do so, we have studied the execution process of an agent and the interactions between these agents in order to better adapt it to our work. This allowed us to launch several agents in parallel on a machine by preparing them for the execution of distributed exclusion algorithms, and also to propose message formats allowing an efficient communication of these agents during the execution of the distributed exclusion algorithms. And finally, we have defined data structures for communication network topologies.

4.2 Communication network topologies

JADE has a very precise architecture allowing the so-called" standardized" construction of agents in accordance with the standards proposed by FIPA and includes all the mandatory components that control an SMA. These components are: the Agent Management System (AMS) which is the agent responsible for managing agent life cycles, the Directory Facilitator (DF) agent which registers the services offered by the agents and the Message Transport service (MTS) agent, also called Agent Communication Channel (ACC), which controls all message exchanges on the platform[8].

However, the AMS, which holds the list of the platforms agent identifiers, is limited when a large number of agents must go through it to get the identifier of an agent with whom the agent wants to communicate. In addition, if we need to register an agent at the DF level in order to find him according to his description when we need his identifier, it

1 http://grail.sdsc.edu/projects/simgrid/

is limited when we have several requests. Because of the diversity of its limitations that we encountered, it was interesting to propose a data structure that would allow our agents to create a network topology that would allow us to efficiently exchange messages without going through AMS and DF. So, for these agents to be able to communicate by exchanging messages, each agent must know the address or identifier of the person they want to communicate with. This has enabled us to create a data structure at the level of each agent that allows us to save the agents' identifiers. The existence of this data structure is important for each agent because it allows the agent to know all the agents in the system without asking AMS. It also avoids overloading the AMS with requests when the number of requests increases.

4.3 Agent communication

As mentioned above, our approach aims to propose an agent-based module for the simulation of distributed mutual exclusion algorithms, the most important step is the synchronisation of the execution of nodes because the problem of mutual exclusion is at the level of competition to an on-shared or critical resource. Thus, synchronising access to this resource can be done by communication between the different processes in a distributed system. Indeed, the messages exchanged by JADE agents have a format specified by the ACL language defined by FIPA. This format lacks a number of parameters, in particular those that did not allow the agents to efficiently synchronise access to a non-sharable resource. At this stage, we encountered difficulties in representing the different messages exchanged between agents by dating the message.

To solve this problem, we redefined the ACLMessage class because all messages in JADE are instances of this class. This allowed us to have a fairly complete message format between the nodes, allowing them to guarantee the proper functioning of the algorithm. The message format thus adopted between the agents is presented as follows:

• Sender-ID: This information makes it possible to know at any time who sent the message;

• Recipient ID: This field allows you to know to whom the message will be delivered;

• The subject of the message;

• The Time-stamp: this is the logical time of message transmission relative to the clock as defined by LAMPORT.

4.4 General architecture of the approach

Our work consists in proposing an agent-based solution for the execution of distributed mutual exclusion algorithms. Distributed mutual exclusion algorithms are executed according to defined network topologies (implementation). 1. For a decentralised network topology, our system consists of n nodes where each node can have the identifiers of the other nodes. These agents are created and registered at the AMS level by an agent named" Network Agent" as shown in figure 1. The created agents execute the same code. System nodes use the same message format to communicate effectively.

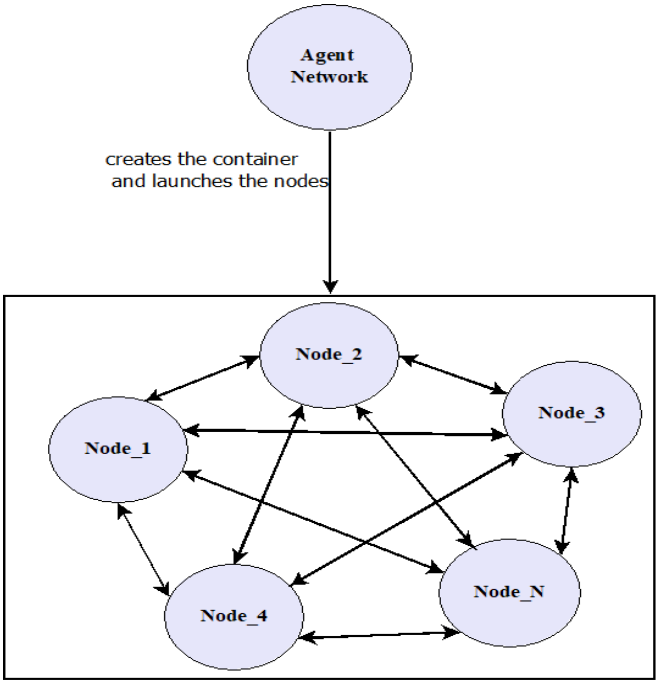

Fig. 1 Decentralised network topology

2. For a hierarchical network topology, our system consists of n nodes, where each node has the identifiers of its group (the child nodes). To define the hierarchical topology, we first have the" network agent" which launches the" root" agent. This created agent launches the other agents named" Parent Agent" and coordinates the opening and closing of sessions when running distributed group mutual exclusion algorithms. Then, the" Parent Agents" launch the nodes and coordinate locally the use of a session. Each node has the identifier of the parent agent and the other nodes. The figure 2 shows the hierarchical network topology between the system's agents.

5. Simulations and experimental results

5.1 Simulation

In order to test that the distributed mutual exclusion algorithms can be simulated on the Jade multi-agent platform, two specific algorithms have been implemented: the Maekawa mutual exclusion algorithm and the *TBGMEAC*$_\alpha$ algorithm.

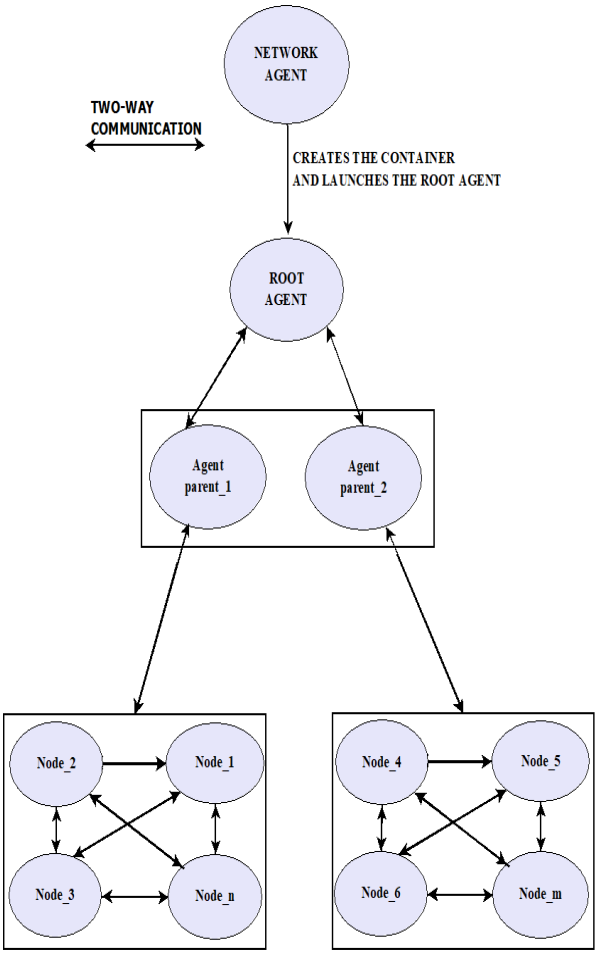

Fig. 2 Hierarchical network topology

Maekawa's mutual exclusion algorithm is chosen for execution on a distributed network topology because it provides a robust system. And the *TBGMEAC*α algorithm is chosen for execution on a hierarchical network topology because hierarchical distributed systems are an important class of distributed systems [4]. The mutual exclusion algorithm is a situation in which, when a process Pi wants to enter a critical section, it sends a Request message to a set of processes Rpi. Upon reception of Rpi — Authorisation messages, process Pi accesses the critical section. When it leaves the critical section, it sends a Release message to the processes of the Rpi set in order to give them back the permissions obtained. A process assigns only one authorisation at a time. When a process Pj receives the request from process Pi, it answers it favorably (by sending an Authorisation message) if it has not already given its authorisation to another Process, or unfavorably (by sending a Failure message) if it has already given its permission to a request from a process Pk older than that of process Pi. When the request from process Pk is older than the one received from process Pi, process Pj sends to process Pka Query message in order to know if it managed to access the critical section or not. When receiving the Polling message, process Pk informs process Pi by sending a Relax message if it has not managed to enter the critical section: this means that it has already received a Failure message.

The figure 3 below shows us an extract of the data resulting from the execution of the Maekawa mutual exclusion algorithm by the agents on JADE.

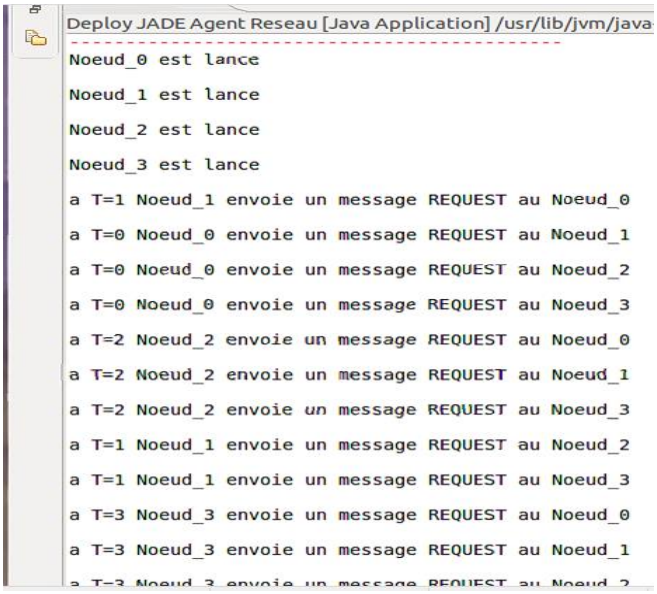

Fig. 3 Result execution of the Maekawa algorithm

The algorithm *TBGMEACα* (Tree Based GME Algorithm for Clusters)

It's an algorithm based on that of Beauquier et al [6]. It uses the technique of partial network flooding to guarantee the EMG. The algorithm is in two phases: opening and closing of a session. These phases are initiated by the root. Indeed, when a process p makes a first request (the very first in the system) for a session X, it is sent to its cluster leader via the ASK (X) message. The leader saves the X session as the requested session, and sends the request to the higher-level cluster leader; thus, from leader to leader the request eventually reaches the root. As soon as the root receives the ASK (X), it begins the operation of opening X by sending OS(X) not only to the process that originated the ASK(X), but also to any other leaders that have not received any requests from their descendants. We call this technique partial flooding. In the *TBGMEACα* algorithm, it is a question of flooding the network with information, whether it is for the opening or closing of a session; this way of doing is significant at the opening of a session in the sense that, as long as no process wants to access a different session from the current one, the processes wishing to use the current session can enter and exit the critical session concurrently and as many times as they wish without generating additional costs to the algorithm. This is because each of them has registered a particular session (e.g. X) as the current session the gold of the login.

Simulation results

In the case of the Maekawa exclusion algorithm, each node (agent) executes the code (program) and communication is done by exchanging messages. **The tables 1 and 2** below present the simulation results of Maekawa's mutual exclusion algorithm by agents (nodes). In the case presented below four agents (nodes) were used to run the Maekawa algorithm.

Table 3 below shows the result of the simulation of the *TBGMEACα* algorithm. The algorithm is deployed on a three-level hierarchical network. We have the root which is the parent of Node 0 and Node 1. Node 0 is the parent of the nodes: Node 00, Node 01, Node 02. And Node 1 is the parent of the nodes: Node 10, Node 11, Node 12.

Table 1: Result of the execution of the Maekawa mutual exclusion algorithm

Nodes	Request	Reply	Fail	Inquire	Relinquish	Critical section	Release
Node0	T=0 Node1 Node2 Node3	Node1	Node1 Node3	Tϵ[4,5] Node 0	Node1 Node2 Node3	Node0	T=0 Node1 Node2 Node3
Node1	T=1 Node0 Node2 Node3	Node0	Node3 Node2				
Node2	T=3 Node0 Node1 Node3	Node3 Node0	Node2 Node1	Node3	Node3		
Node3	T=2 Node0 Node1 Node2	Node2 Node0	Node2 Node1	Node2	Node2		
After the critical section and sending release messages, from the Node 0							
Nodes	Request	Reply	Fail	Inquire	Relinquish	Critical section	Release
Node0	T=6 Node1 Node2 Node 3						
Node 1	Node 3	Tϵ[7,8] Node 1	Node0 Node3 Node2	Node1	Node3	Tϵ[7,8] Node1	Node0 Node3 Node2
Node2	Node1						
Node3	Node1						

Table 2: Result of the execution of the Maekawa mutual exclusion algorithm

After the critical section and the sending of release messages, from the Node1							
Nodes	Request	Reply	Fail	Inquire	Relinquish	Critical section	Release
Node0		Node3					T=0 Node1 Node2 Node3
Node 1	T=9 Node0 Node2 Node3						
Node2	Node3						
Node 3	Node 2				Tϵ[10,11] Node 3	Node0 Node1 Node2	Node0 Node1 Node2
After the critical section and the sending of release messages, from the Node 3							
Nodes	Request	Reply	Fail	Inquire	Relinquish	critical section	Release
Node0							
Node1		Node2					
Node 2						Tϵ[13,14] Node2	Node0 Node1 Node3
Node3	T=12 Node0 Node1 Node3						

Table 3: Result of the execution of the *TBGMEACα* algorithm

Nodes	Send ASK(0)	Send Request session (0)	Send OS (0)	critical Section	Receive OS (0)	Receive Request session (0)	receive ASK (0)
Node 00	T=0 Node 0			Tϵ[1,2] Node 00	Node 0		
Node 0		Root	Node00 Node01 Node 02				Node00 Node 02
Root			Node0 Node 1			Node 0	
Node 1			Node10 Node11 Node 12				
Node 02	T=3 Node 0			Tϵ[4,5] Node 02			

5.2 Discussion

Looking at the tables 1 and 2, obtained from the result of the execution of Maekawa's mutual exclusion algorithm, we also notice that, each node having requested access to a critical section in a given time succeeds in accessing it when the critical section is free. This means that vivacity (a property of mutual exclusion algorithms) is ensured during the execution of the Maekawa mutual exclusion algorithm by the nodes because:

• At T= 0, Node 0 sent a REQUEST message to the nodes (Node 1, Node 2, Node 3) but enters the critical section at T= 4 and leaves the critical section at T= 5.

• At T=1, Node 1 sent a REQUEST message to the nodes (Node 0, Node 2, Node 3) but enters critical section at T=7 and leaves critical section at T=8.

• At T= 3, Node 2 sent a REQUEST message to the nodes (Node 0, Node 1, Node 3) but enters the critical section at T=13 and leaves the critical section at T=14.

• At T=2, Node 3 sent a REQUEST message to the nodes (Node 0, Node 1, Node 2) but enters critical section at T=10 and leaves critical section at T=11.

One other remark can be made. Indeed, in no case could we have more than one node in critical section (executing the same part of non-sharable code) at a given time during the execution of Maekawa's mutual exclusion algorithm. This is because the time (Tϵ[4.5]) when Node 0 executes the critical section is different from the time (Tϵ [7.8]) of Node 1, and is also different from the times Tϵ [10.11] and Tϵ [13.14] respectively of Nodes 2 and 3 when they were executing the critical section. This means that safety is assured. It appears that the proposed agent-based approach is important for evaluating mutual exclusion algorithms since the agents execute the algorithm in a way that satisfies the mutual exclusion properties.

Looking at the table 3 From the result of the execution of the *TBGMEACα* algorithm, we notice that the execution of the described *TBGMEACα* algorithm is ensured because when a child node requests the opening of a section 0, it sends the request to the higher level cluster leader; thus, from leader to leader the request ends up reaching the root. This is the case of Node 00 in the table 3. Node 00 sends an ASK (0) message to its parent Node 0 to ask it to log in 0. Node 0 in turn sends a Request session (0) message to the root that is its parent.

We notice again that the root opens the requested session when it receives the message from its children via an OS message; and those children forward the message to their descendants. This is the case of the root in the table 3 which sends the message OS (0) to his sons Node 0 and Node 1, who in turn send Node 00, Node 01, Node 02 and Node 10, Node11, Node 12 to their respective descendants. Finally, we note that if the nodes want to access a session, and no other node wants to access a different session, then these nodes can enter it concurrently. This is the case of Node 00, and Node 02 in the table 3. Hence the concurrent entry property (Efficiency) of group mutual exclusion algorithms is ensured. And if a node wants to access a session 0, then that node finally executes its critical section. This is the case of Node00 which solicited session 0. Hence the absence of blocking and consequently the property of Assured Vivacity. And also, in no case could two sessions of different open groups be observed at the same

time: this shows the Security. This is the case of session 0, which is the only one open when the Node has solicited. From this discussion we can conclude that, it is possible to simulate the distributed algorithms of mutual exclusion on JADE in order to better analyse the results.

6. Conclusion and future work

The objective of this work was propose from the JADE multi-agent framework, an agent-based module for the simulation of distributed mutual exclusion algorithms. JADE is a simple and easy to use platform. It allows to implement a multi-agent system in a way that the agent behavior is easily understandable, modifiable, and is reasonable for implementing message-oriented applications that can be applied in reality. This flexibility allowed us to study the execution process of an agent and the interactions between these agents. On the basis of these studies, we proposed a module allowing to launch several agents in parallel on a machine by preparing them for the execution of distributed exclusion algorithms. We have also implemented and simulated two classical and group distributed mutual exclusion algorithms to verify the properties of these mutual exclusion algorithms.

Our results should help researchers in the study of distributed mutual exclusion algorithms to perform their simulation on the JADE platform. The results obtained so far show that it is now possible to test or simulate the distributed mutual exclusion algorithm on the multi-agent JADE platform.

In perspective, adjustments can still be made on the implementation of the distributed mutual exclusion algorithms from the JADE platform. We will study the behavior of JADE in terms of migration of agents for simulation in a dynamic system.

References

[1]http://nchc.dl.sourceforge.net/sourceforge/nsnam/.
[2]http://www-verimag.imag.fr/˜devismes/sinalgo/.
[3]http://www.risc.uni-linz.ac.at/software/daj/.
[4] Swaroop A. Efficient group mutual exclusion protocols for message passing distributed computing systems. PhD thesis, National Institute of technology, Kurukshetra, Haryana, India, PIN-136119, october2009.
[5] Fabien BADEIG. Un environnement actif pour la simulation multiagents. PhD thesis, Universite Paris Dauphine, septembre 2010.
[6] Datta A. Beauquier J., Cantarell S. and Petit F. Group mutual exclusion in tree networks. Journal of Information Science and Engineering, Vol.19: pp. 415–432, 2003.
[7] Lars Lundberg Paul Davidsson Dawit Mengistu, Peter Troger. Scalability in distributed multi-agent based simulations:the jade case. Second International Conference on Future Generation Communication and Networking Symposia, 2008.
[8] Tiziana Trucco Giovanni Rimassa Fabio Bellifemine, Giovanni Caire. JADE PROGRAMMER'S GUIDE. TILAB, formerly CSELT, University of Parma, 08-April-2010.
[9] Ferber. Les systemes multi-agents, vers une intelligence collective. Inter Editions,1995.
[10] H.Casanova. Simgrid: a toolkit for the simulation of application scheduling. Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2001), Brisbane, Australia, pages 15–18, May 2001.
[11] ENOUDINA LAZHAR. Modélisation et simulation basees multi-agents du contrôle de processus industriels. phd thesis, universite 20 aout 1955skikda, 2009.
[12] Julien Sopena. Algorithmes d'exclusion mutuelle : tolérance aux fautes et adaptation aux grilles. PhD thesis, December 2008.

M. SENOUA César is a researcher in computer engineering. He is currently a researcher at the National Institute of Cartography/MINRESI, Cameroon. He obtained his master's degree in 2014 from the University of Ngaoundere in Cameroon. His current research focuses on distributed systems, geographic information systems, multi-agent systems.

Dr. KAMLA Vivient Corneille is a Lecturer in the Department of mathematics and computer science in the National school of agroindustrial sciences, University of Ngaoundere, Cameroon. He is a holder of a Ph.D degree in Applied Mathematics in 2008 from the University of Yaounde 1 in Cameroon and the University of Pau and Pays de l'Adour (UPPA), in an international joint supervision. His current research interests include multi-agent systems, distributed systems, simulation and modeling.

Pr. Blaise Omer YENKE is a Senior Lecturer and researcher in computer engineering. He is the head of Department of computer engineering at the University Institute of Technology, University of Ngaoundere, Cameroon. He obtained his Ph.D. degree in computer science in 2010 from the University of Yaounde 1 in Cameroon and the University of Grenoble in France, in an international joint supervision. His current research interests include HPC, distributed systems, fault tolerance, sensor networks design and sensor's architecture.

Pr. KAMGANG Jean Claude is a Senior Lecturer and researcher in mathematic. He is the head of Department of mathematics and computer sciences in the National school of agroindustrial sciences University of Ngaoundere, Cameroon. His current research interests include Epidemiology Mathematics, control, dynamic systems, modeling and analysis of dynamics.