



Improving Fault Tolerance Mechanisms Implemented by WADE Platform

ADEDJOUMA Semiyoun A. and Brice X. R. HESSOU

Laboratory of Electro-technical, Telecommunications and Applied Computing (LETIA) of the Polytechnic School of Abomey-Calavi (EPAC),
University of Abomey-Calavi (UAC), Republic of BENIN

Available online at: www.isca.in

Received 27th February 2013, revised 12th March 2013, accepted 10th April 2013

Abstract

Problems related to scalability, reliability, protection of privacy and energy consumption have led to the development of distributed architectures for the workflows execution including WADE platform (Workflows and Agents Development Environment). However, further problems remain, such as those related to network connectivity. It has therefore become necessary to make these architectures fault tolerant. In this paper, we improved fault tolerance mechanisms implemented by WADE platform which is an agent-based workflow engine. To solve the problems of scalability and maintainability of the platform revealed by fault tolerance mechanisms implemented by WADE, we proposed an additional functionality based on the distributed storage concept of Chord protocol. Unlike existing mechanisms, our proposal has helped to restart all agents of a container which has suddenly died.

Keywords: Fault tolerance, workflow, WADE, distributed architecture, Chord.

Introduction

Workflow is a key technology that supports and automates business processes. Workflow management is ensured by specific systems called Workflow Management Systems. These systems usually rely on a centralized coordinator (motor) knowing all the information of flow control and data workflow, thus posing problems not only in scalability, reliability, but also of protection privacy or energy consumption¹. It has therefore become necessary to develop decentralized alternative architectures for that coordination. Among these architectures, the platform Workflows and Agents Development Environment (WADE) is an agent-based workflow engine². It is based on the "framework" JADE, a software platform for the development of multi-agent systems (MAS)³.

However, the unavailability of a host harboring a service can lead to the loss of the workflow, which would raise system performance problems. To solve this issue, fault tolerance has been integrated into decentralized architectures for the execution of workflows. Indeed, the fault tolerance property is defined by the ability of the system to maintain its functionality, in case of failure of some of its components. It aims to minimize the impact of these failures on the overall task of the system⁴.

Therefore, the WADE platform also incorporates mechanisms for fault tolerance. It inherits from JADE fault tolerance mechanisms such as the replication of the "Main Container". These mechanisms provided by JADE, are only involved at the platform level. WADE implements other specific mechanisms instead involved in application level⁵. Indeed, these mechanisms ensure an automatic restart of all agents which suddenly die. In order to achieve this, each container (except the "Main Container") hosts a special agent called "Control Agent" (CA)

that is responsible for the supervision of all the resources of the container and the automatic restart of a local agent. All CAs agree to elect a leader. The automatic restart mechanism works as follows⁵: i. If an agent dies, the local CA restarts it. ii. If a whole container dies, causing the death of its CA, the leader CA carries out restarting the container and all its agents. iii. If the dead container nests the leader CA, the other CAs agree to re-elect a new leader which then restart the container and all its agents. iv. If the restarting of an entire container fails (which usually happens when the underlying host is unreachable), the leader CA will then via the RAA (Runtime Allocator Agent), recreate all dead agents.

Indeed, the RAA has a list of agents allocation policies which are all contained in a configuration file and it reads this file at the startup of the platform. This means that before integrating or removing a new container or a new host, the allocation policies should be redefined in the file and the whole platform restarted if they must survive to faults. This approach thus confronts us to the problems of maintainability and scalability of WADE applications. We propose in this paper an improvement of fault tolerance mechanisms implemented by the platform WADE.

In Section 2 we present our proposal specifically through its modeling and its working principle and the adopted methodology. Then, in Section 3, we present the results obtained which are analysed in Section 4.

Material and Methods

Architecture preexisting: The autorestart process is implemented by an instance of the class Container Restarter which is an extension of the class FMS Behaviour, behaviour

defined as a state machine. Figure-1 shows the state machine which outlines the process for a container restart.

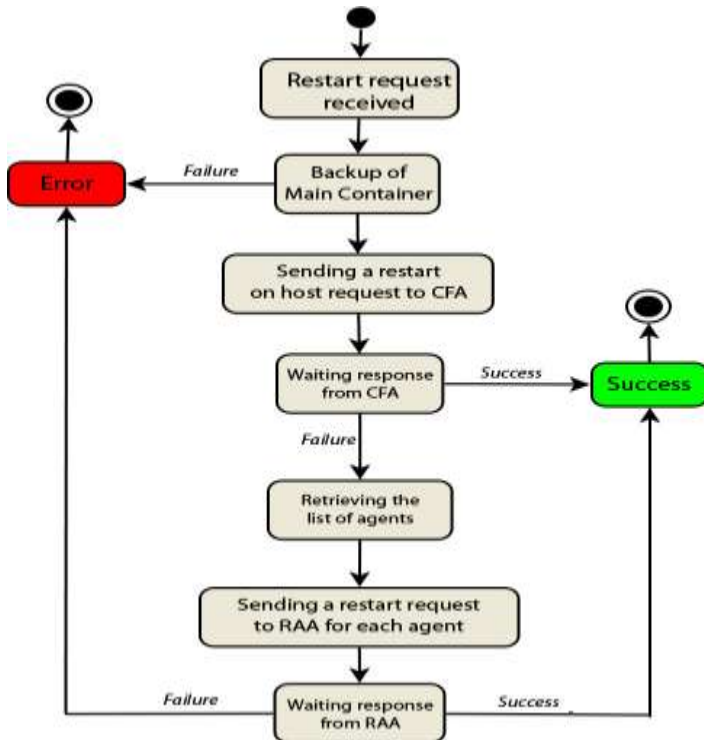


Figure-1
State machine for starting a container

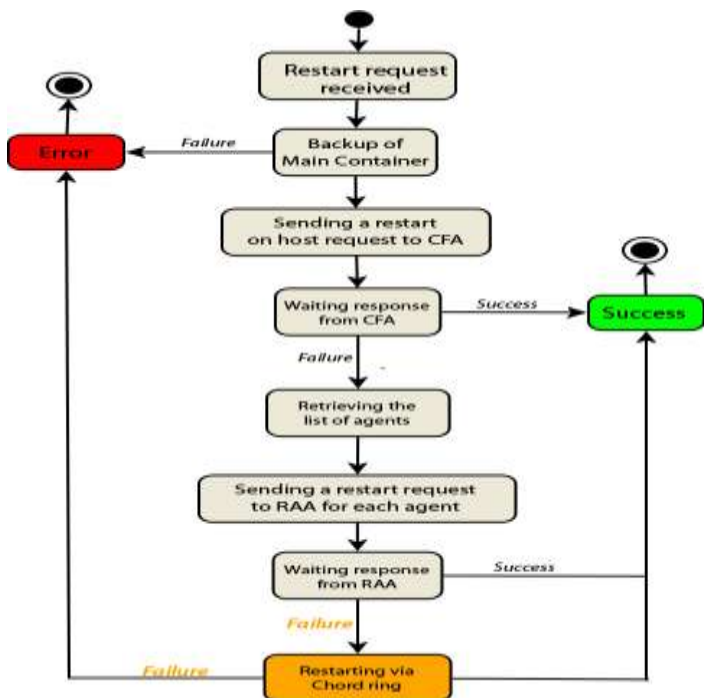


Figure-2
State machine of our proposal

In fact, when a container dies, the leader CA first attempts to restart the host nesting it. In case of failure, it determines the list of all agents nested by the container and asks the RAA to restart these agents depending on the allocation policies defined by the developer. If no allocation policy corresponds to the agent, the RAA returns a restart failure message to the leader CA and the agent concerned is no longer restarted.

The leader CA will use a new step figure-2 that we have added to the self-repair process of WADE when it receives a failure from the RAA, that is, when no allocation policy allows to select a container able to nest the living agent.

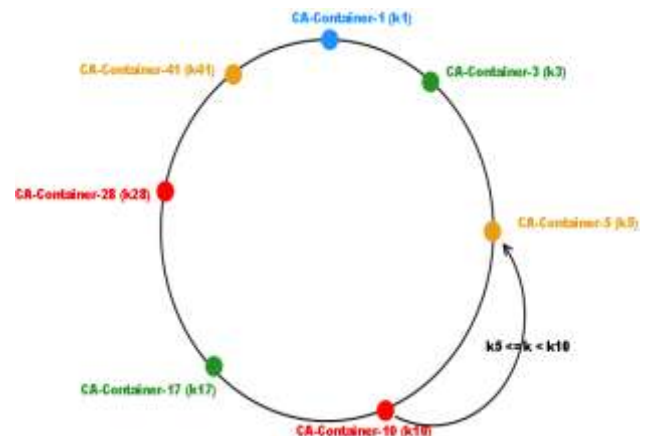


Figure 3
Structure of the Chord ring obtained

Modeling of our solution: Allocation policies allow the RAA to select a host or container which will nest the agent to restart. This means that when no container has been selected, recreating the agent fails. Thus, to improve this mechanism, the selection of a living container or host to nest the agent to recreate should be ensured, regardless the allocation policies. We have therefore proposed an additional step in the container autorestart process which is based on the principle of distributed data storage provided by the Chord protocol⁶.

During this step, the CAs are the nodes of our ring in figure-3. The keys are obtained using the hash function SHA-1 which takes as a parameter the identifier of the agent that is its name.

Operating principle of our solution: Our ring has allowed us to determine the container that will nest the agent to restart. Thus, we sought in the ring, the node responsible for a given key (the one corresponding to the agent to restart). This node corresponds to a CA from which we obtained all the necessary information on its container.

Once the information related to container obtained, the leader CA can then restart the agent via the AMS. Figure-4 shows a sequence diagram summarizing the operating principle of our solution

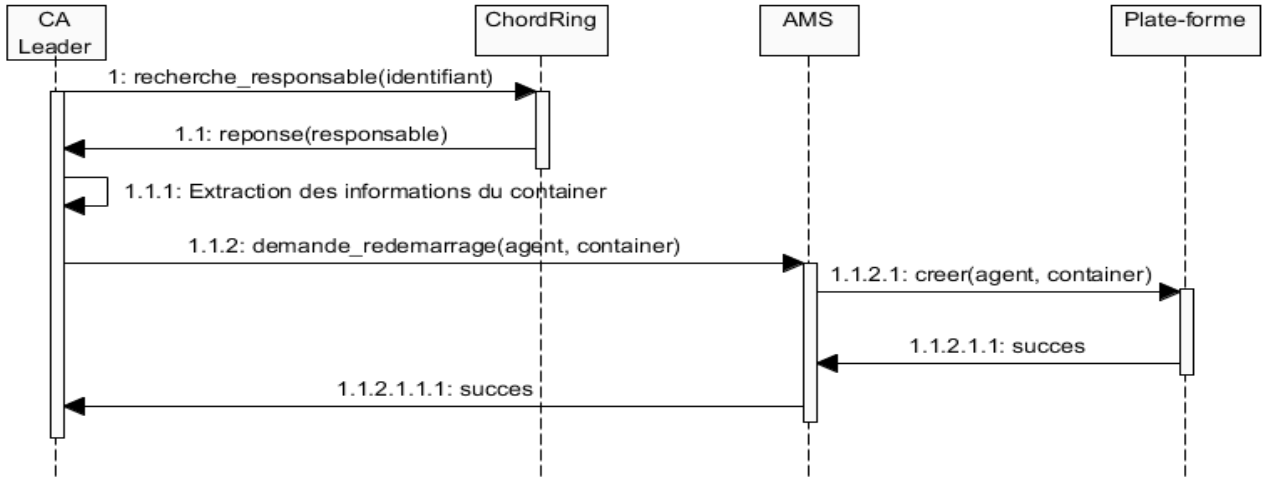


Figure-4
Sequence diagram of our solution

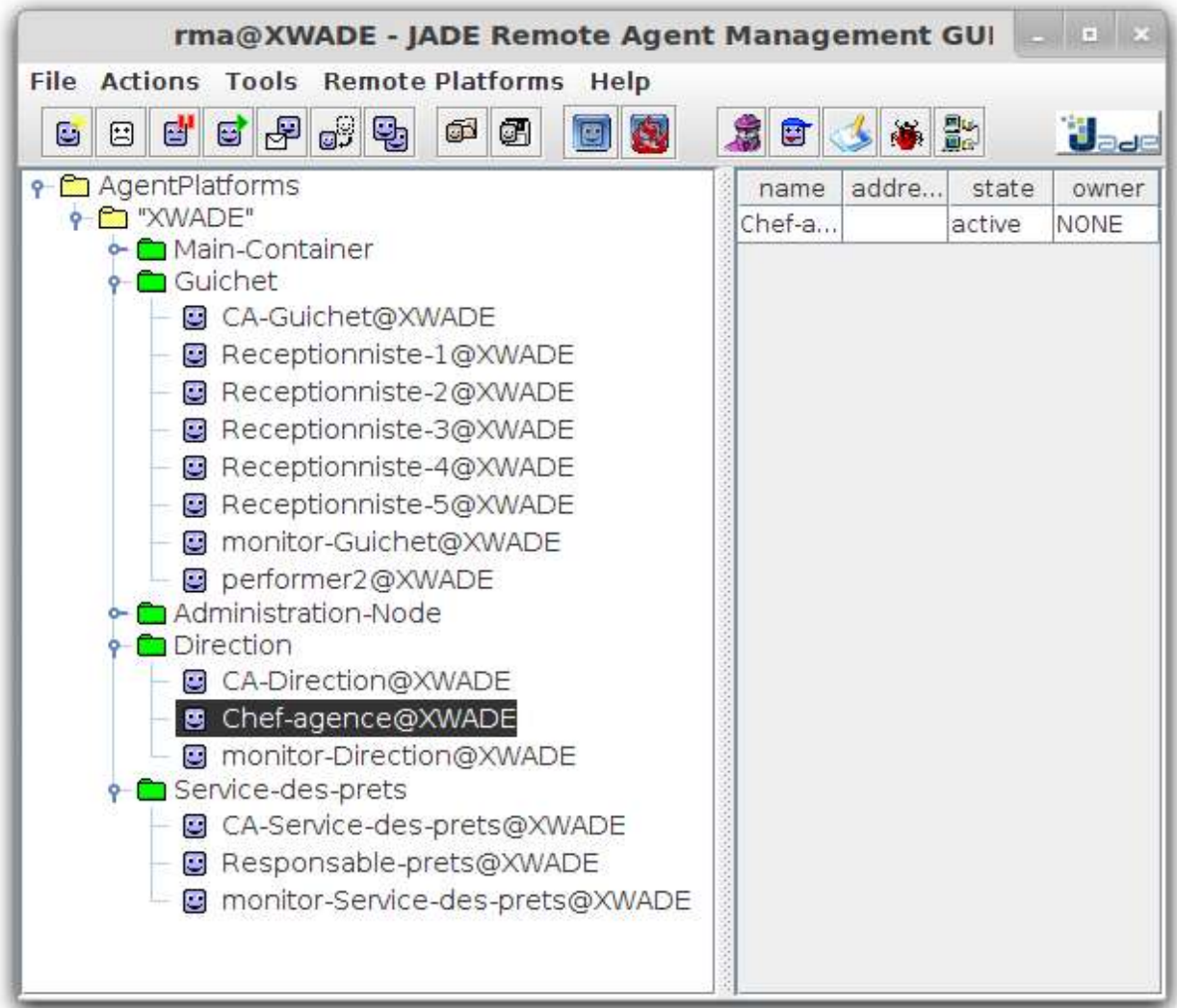


Figure-5
Configuration of the platform at the start of the application of bank loan process workflow

Results and Discussion

Results: We added a new step to the autorestart process. Indeed, some WADE classes such as "ControllerAgent" (representing the CAs), "ContainerRestarter" (which implements the autorestart process), etc. have been modified. We also added new classes and new packages ("com.tilab.xwade.ca" and "com.tilab.xwade.chord" packages). The package "com.tilab.xwade.chord" has been used to implement the distributed storage principle of the Chord protocol. The class "StartAgentsViaRing" of the package "com.tilab.xwade.ca" has been used to restart the agent via our Chord ring.

To evaluate the performance of our proposal, we performed simulations using different workflow applications.

Workflow of the bank loan process: The workflow is widely used in the field of finance. Specifically, it is used to automate

the granting of loans. Hence, one of the examples used illustrates this process. This workflow is used to model the process from the receipt of the loan application to its granting or rejection. For this application, we have not defined allocation policies that correspond to the various agents used.

We executed this example on native WADE, then on modified WADE in figure-5. After launching the application, we removed from the network, in each case, the host nesting the agent named "Chef-agence". With the native platform WADE, the latter has not been restarted, as depicted by the traces of the log file obtained figure-6 because no allocation policy has been defined for this agent. With modified WADE this agent has been restarted on the container "Services-des-prets", see figure-7 and figure-8, even if no allocation policy matches with it. This shows the effectiveness of the autorestart process of our proposal compared to native WADE.

```
143WADE: Container Direction unexpectedly dead! [com.tilab.wade.ca.ControllerAgent handleRemovedContai
144orestart active
145m the leader. I reboot the container
146E: Activating restart procedure for container Direction... [com.tilab.wade.ca.ControllerAgent$9 onW
147E: Requesting cfa to restart container Direction in host host-5 [com.tilab.wade.ca.ContainerRestart
148ADE: FAILURE message received as response from CFA to StartContainer request about container Direc
149E: Try to restart agents of container Direction using RAA [com.tilab.wade.ca.ContainerRestarter$Sta
150ADE: FAILURE message received as response from RAA to CreateAgent request about agent Chef-agence.
151E: Control agent CA-Journaliste-Culture removed. [com.tilab.wade.ca.ControllerAgent handleDeadAgent
152tainer Journaliste-Culture removed (autorestart=true, leadership=true)
153WADE: Container Journaliste-Culture unexpectedly dead! [com.tilab.wade.ca.ControllerAgent handleRem
154orestart active
155. I am the leader. I reboot the container
```

Figure 6
Traces of the WADE log file for the application of the bank loan process workflow

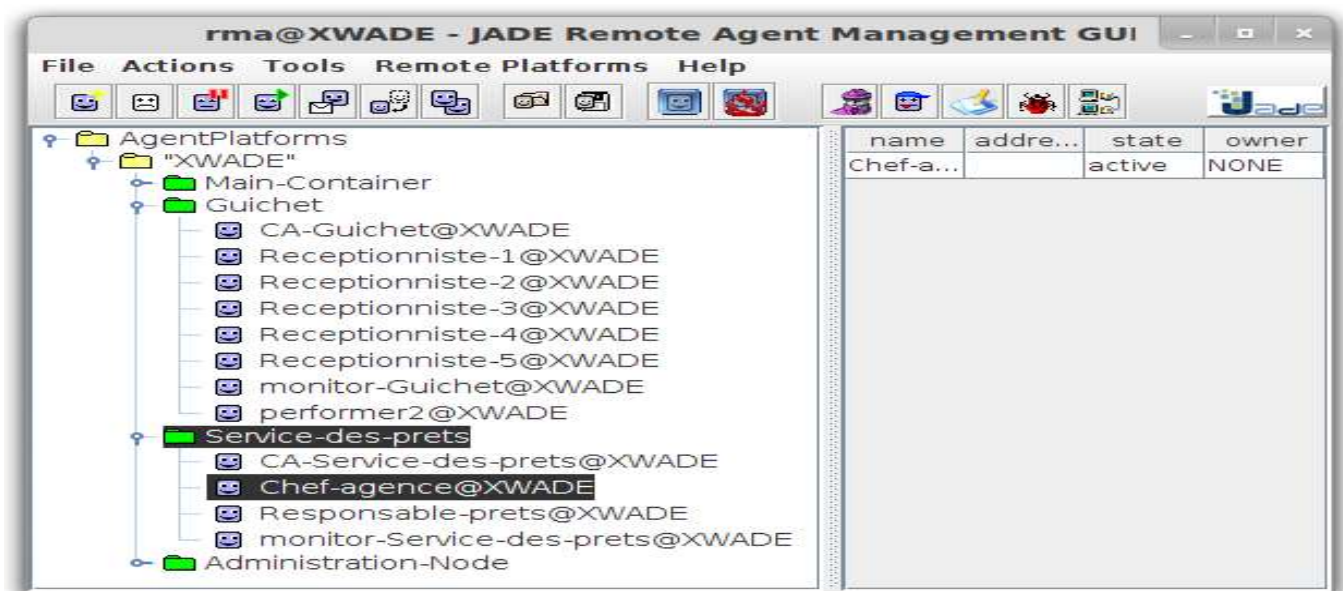


Figure 7
Configuration of the platform after the removal of the host nesting the container "Direction" (with modified WADE platform)

```

143XWADE: Container Direction unexpectedly dead! [com.tilab.wade.ca.ControllerAgent
144torestart active
145DE: 6 CAs has been found. [com.tilab.wade.ca.ControllerAgent makeRing()]
146DE: Chord ring has been established successfully. [com.tilab.wade.ca.ControllerA
147am the leader. I reboot the container
148DE: Finger tables have been fixed successfully. [com.tilab.wade.ca.ControllerAge
149DE: Activating restart procedure for container Direction... [com.tilab.wade.ca.C
150DE: Requesting cfa to restart container Direction in host host-5 [com.tilab.wade
151WADE: FAILURE message received as response from CFA to StartContainer request at
152DE: Try to restart agents of container Direction using RAA [com.tilab.wade.ca.Co
153WADE: FAILURE message received as response from RAA to CreateAgent request about
154DE: Try to restart agent Chef-agence using Chord ring. [com.tilab.xwade.ca.Start
155DE: Re-creating agent Chef-agence using Chord ring. [com.tilab.xwade.ca.StartAge
156DE: Try to get the ring's node which is responsible of agent Chef-agence. [com.t
157DE: Container Services-des-prets has been selected to nest agent Chef-agence. [c
158DE: Agent Chef-agence successfully re-created [com.tilab.xwade.ca.StartAgentsVia
159DE: Control agent CA-Journaliste-Culture removed [com.tilab.wade.ca.ControllerA

```

Figure 8

Traces of the log file of the amended WADE for the application of the bank loan process workflow

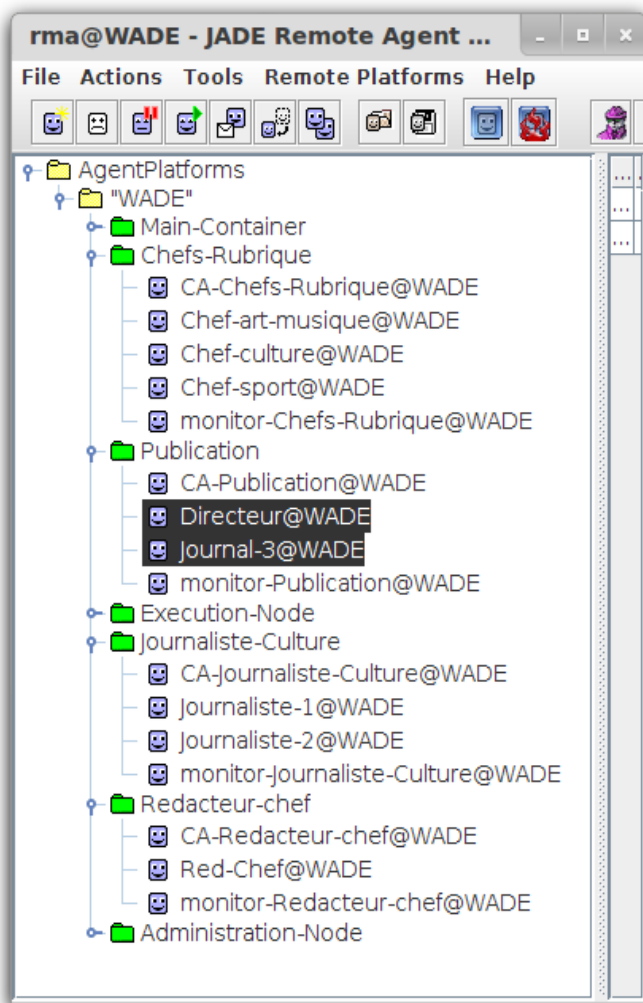


Figure 9

Configuration of the platform to start the application of the newspaper article writing workflow

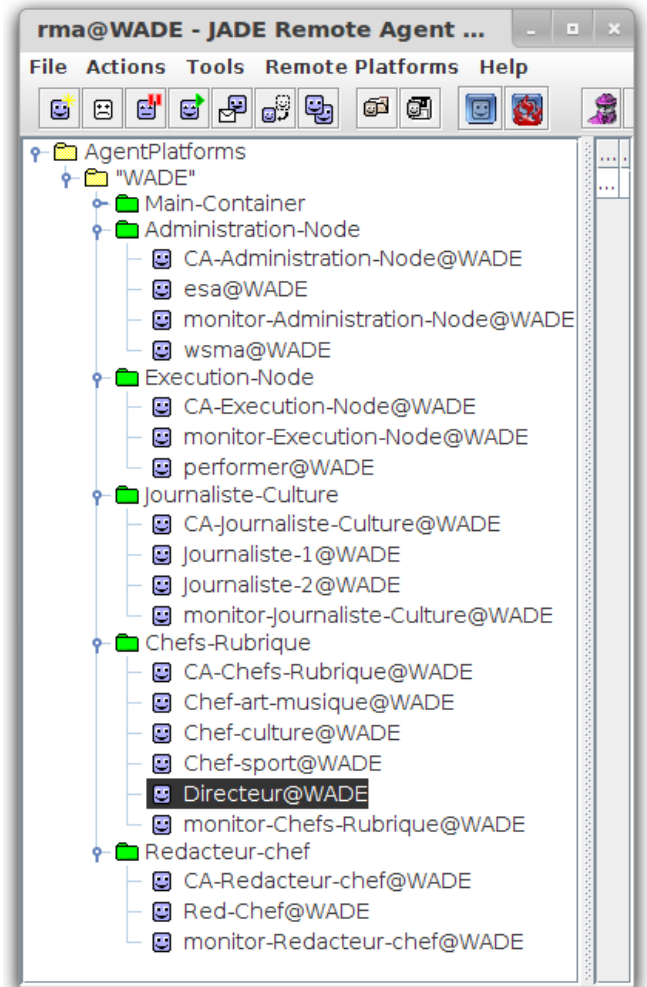


Figure 10

Configuration of the platform after the removal of the host nesting the container "Publication" (with native WADE)

Workflow for writing a newspaper article: This application illustrates the process of writing and publishing a newspaper article. We implemented the workflow by considering four (04) types of actors, namely: the Publication Director ("Directeur-de-publication"); the Editor ("Rédacteur-en-chef"); the Head section ("Chef-rubrique"); the Journalist ("Journaliste"). For this application, we defined a single allocation policy in addition to the default which corresponds to the agents of type "Directeur-de-publication".

We also executed this application first on the native platform WADE then on the modified WADE. After starting the

application **Figure-9**, we have suddenly removed from the network, the host nesting the container "Publication". With the native platform WADE, only the agent "Director" has been restarted. The restart of the agent "Journal-3" failed because no allocation policy fit it in **Figure-10** and **Figure-11**. However, with the modified WADE all the missing agents were successfully restarted in **Figure-12** and **Figure-13**. This also shows that performance of the self-healing mechanism of the modified WADE platform is better than that of the native WADE.

```

143:IALE: Container Publication unexpectedly dead! [com.tilab.wade.ca.ControllerAgent handleRemovedCon
144:to restart active
145:LE: Activating restart procedure for container Publication... [com.tilab.wade.ca.ControllerAgent$9
146:LE: Requesting cfa to restart container Publication in host letia-005 [com.tilab.wade.ca.ContainerR
147:IALE: FAILURE message received as response from CFA to StartContainer request about container Publ
148:LE: Try to restart agents of container Publication using RAA [com.tilab.wade.ca.ContainerRestarters$
149:LE: CreateAgent request received from raa@WADE. Name = Directeur class = editoriale.acteurs.DPAgen
150:LE: Local agent Directeur@WADE. successfully started [com.tilab.wade.ca.ControllerAgent serveCreate
151:Starting..... [com.tilab.wade.commons.WadeAgentImpl setup()]
152:IALE: FAILURE message received as response from RAA to CreateAgent request about agent Journal-3 .
153:STARTUP COMPLETED SUCCESSFULLY! [com.tilab.wade.commons.WadeAgentImpl setup()]
154:LE: Control agent CA-Journaliste-Culture removed. [com.tilab.wade.ca.ControllerAgent handleDeadAgen

```

Figure-11

Traces of the log file of the native WADE for the application of the newspaper article writing workflow

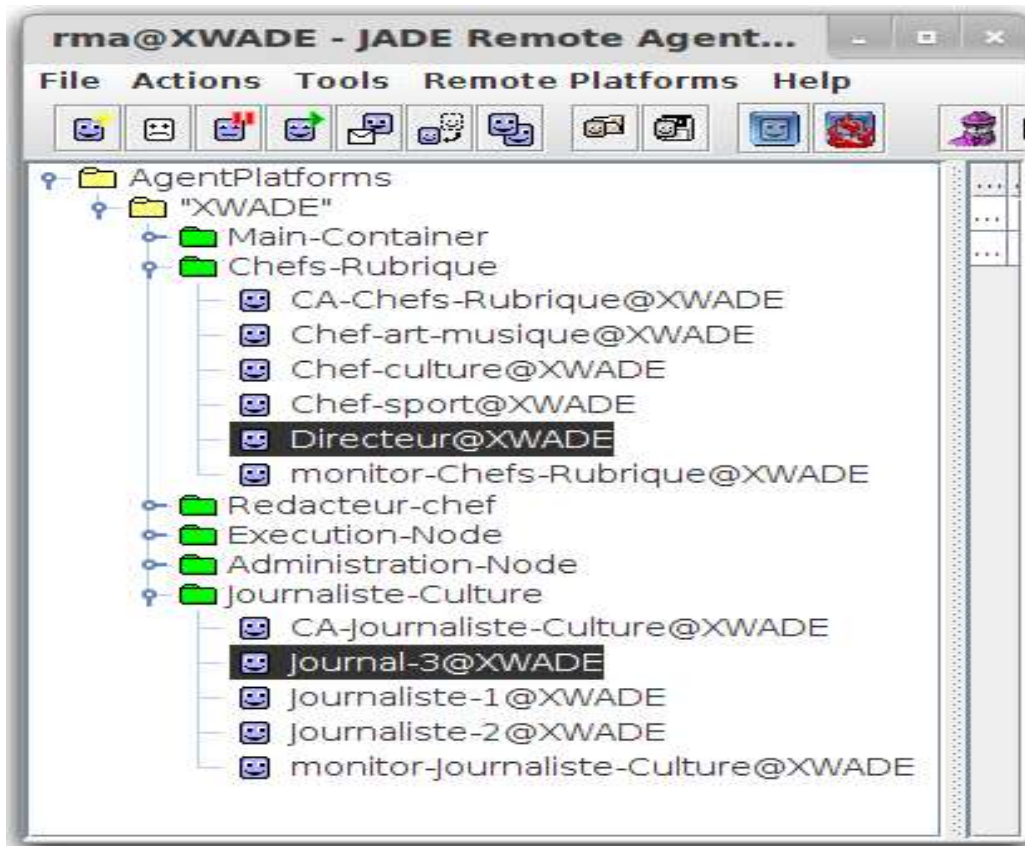


Figure-12

Configuration of the platform after the removal of the host nesting the container "Publication" (with modified WADE)

```
142: Container Publication removed (autorestart=true, leadership=true)
143: E: Container Publication unexpectedly dead! [com.tilab.wade.ca.ControllerAgent
144: restart active
145: 6 CAs has been found. [com.tilab.wade.ca.ControllerAgent makeRing()]
146: Chord ring has been established successfully. [com.tilab.wade.ca.ControllerAgent
147: the leader. I reboot the container
148: Finger tables have been fixed successfully. [com.tilab.wade.ca.ControllerAgent
149: Activating restart procedure for container Publication... [com.tilab.wade.ca.Co
150: Requesting cfa to restart container Publication in host host-5 [com.tilab.wade
151: E: FAILURE message received as response from CFA to StartContainer request about
152: Try to restart agents of container Publication using RAA [com.tilab.wade.ca.Co
153: CreateAgent request received from raa@XWADE. Name = Directeur class = editoria
154: Local agent Directeur@XWADE successfully started [com.tilab.wade.ca.Controller
155: arting..... [com.tilab.wade.common.WadeAgentImpl setup()]
156: E: FAILURE message received as response from RAA to CreateAgent request about a
157: Try to restart agent Journal-3 .using Chord ring. [com.tilab.xwade.ca.StartAge
158: Re-creating agent Journal-3 .using Chord ring. [com.tilab.xwade.ca.StartAgents
159: Try to get the ring's node which is responsible of agent Journal-3 . [com.tila
160: Container Journaliste-Culture has been selected to nest agent Journal-3 . [com
161: Agent Journal-3 .successfully re-created [com.tilab.xwade.ca.StartAgentsViaRing
162: ARTUP COMPLETED SUCCESSFULLY! [com.tilab.wade.common.WadeAgentImpl setup()]
163: Control agent CFA Journaliste-Culture removed [com.tilab.wade.ca.ControllerAgent
```

Figure-13

Traces of the log file of amended WADE for the application of the newspaper article writing workflow

Discussion: The Chord ring allowed us to select a container among those living to nest an agent, unlike allocation policies which correspond to an agent or group of agents⁵.

Our simulations have shown that with the modified WADE platform, all dead agents are restarted, which is not always the case with the native WADE. We note therefore that the fault tolerance mechanisms implemented by the amended WADE might completely replace those implemented by the native WADE.

However, the selection of the candidate container was based on the numeric keys corresponding to the hash values of the agents' names. Therefore, the distribution of dead agents is totally random and depends only on the names of the agents which are chosen by the developer. This might lead us to obtain a distribution not suitable to the specifics of the application or non-optimal. Allocation policies appear to be a solution to this problem.

In addition, the use of the hash function SHA-1 significantly reduces the risk of collisions regarding the numeric keys of the ring and perfectly balances them, thus eliminating any bottleneck throttle at one or more nodes unlike the existing mechanism⁷.

Finally, our solution ensures the scalability and maintainability of the application unlike the autorestart process of the native WADE. Indeed, the establishment of the ring and its maintenance are dynamic and independent of constraints and specificities of the application. This favors the insertion and removal of one or more containers according to the new

requirements of the application and without encroaching on the performance of the self-healing system.

Conclusion

In this paper, we proposed an improved implementation of fault tolerance by the WADE platform. WADE is an agent-based workflow engine used for distributed execution of workflows. To ensure fault tolerance at the application level, WADE provides mechanisms for automatic restart of any container suddenly disappeared from the platform. This mechanism is not effective for all WADE applications. We have therefore proposed an additional step in the container autorestart process. The proposal based on the principle of distributed data storage of the Chord protocol, allow selecting among those living a container to nest the agent to restart. Far from being a substitute for the existing mechanism, our proposal rather strengthens the self-healing mechanism implemented by the WADE platform.

References

1. Alonso G., Mohan C., Agrawal D. and Abbadi A.E., Functionality and limitations of current workflow management systems. *IEEE Expert* 12, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.7824&rep=rep1&type=pdf> (1997)
2. Caire G., Gotta D. and Banzi M., WADE: A software platform to develop mission critical applications exploiting agents and workflows, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) – Industry and Applications Track*, Berger, Burg,

- Nishiyama (eds.), May, 12-16., Estoril, Portugal, 29-36 (2008)
3. Bellifemine F., Caire G., Poggi A. and Rimassa G., JADE a White Paper, *EXP in search of innovation, The Development of the GPRS General Packet radio System Lightweight Extensible Agent Platform*, **03(3)** 6-19 (2003)
 4. Avizienis A., Laprie J.C., Randell B. and Landwehr C., Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on dependable and secure computing*, **01(1)** 11-33 (2004)
 5. Caire G., WADE User Guide, *Telecom Italia S.p.A.*, <http://jade.tilab.com/wade/doc/WADEUser-Guide.pdf> (2011)
 6. Stoica I., Morris R., Karger D., Kaashoek M.F. and Balakrishnan H., Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *In Proc. of ACM SIGCOMM (San Diego, California, Aug. 2001)*, 149-160 (2001)
 7. Feltin G., Doyen G. and Festor O., Les protocoles Peer-to-Peer, leur utilisation et leur détection, <http://2003.jres.org/actes/paper.70.pdf> (2003)