Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 147 (2014) 429 – 435

Procedia
Social and Behavioral Sciences

ICININFO

An Agent Based Architecture Benchmark

Petros Belsis[a]*, Stefanos Gritzalis[b], Costas Lambrinoudakis[c], Christos Skourlas[a], Vassilis Tsoukalas[a], Dimitris Vassis[a]

[a]*Department of Informatics, TEI of Athens, Agiou Spyridonos 28, Aigaleo 12210, Greece*
[b]*Department of Information and Communication Systems Engineering, Univ. of the Aegean, Carlovassi, Samos, 83200, Greece*
[c]*Department of Digital Systems, Univ. of Piraeus, Karaoli and Dimitriou 80, Piraeus,18534, Greece*

Abstract

Agent based platforms provide a means for creating applications that run independently of operating system and network architecture; as a result, agents have become part of many systems and support a large number of interactions between different systems. Lately due to a shift to mobile computing paradigms, lightweight platforms, mainly oriented for resource constrained devices have emerged. To this end it is important to identify the load and to measure the performance of agent systems with appropriate benchmarks that provide details to direct future implementations.

Keywords: Sofware agents; Benchmarking

1. Introduction

In recent years the shift to distributed computing and the development of autonomous platforms for services provision has led to the adoption of several platforms that enable distributed communication in real time. Within the context of distributed computing efficiency is a key property. A popular framework for development of multi-agent systems that builds upon the Java programming environment is the JADE (Java Agent Development Framework) (Bellifemine et al, 2008). JADE is compliant with FIPA standards that define the rules for inter-agent communication in a multi-agent environment, and is used for the simplification of construction of multi-agent systems. A multi-agent system needs to support several requirements (Belsis et al, 2008) (Zafeiris et al, 2005):

* Corresponding author. Tel.: +30-2105910974; fax:+30-2105910975.
E-mail address: pbelsis@cs.teiath.gr

doi:10.1016/j.sbspro.2014.07.134

- Scalability (Turner et al, 2000). The number of the participating domains should be able to grow, without restricting the functionality of the system.
- Transparency. Asset identification as well as security management should be able to be performed with minimal effort on the user's behalf. In fact, the less the user involvement, the higher the user satisfaction when intelligent interfaces provide the ability to carry out the user assigned tasks within the distributed environment.
- Security management. In fact, the system should retain all the security properties for each separate domain, plus it should allow all accesses within the federated environment that will be defined as permitted by the system's administrators.
- Heterogeneity management. A main obstacle to be overcome is the fact that different organizations use different structural as well as data encoding representations; thus, overcoming this problem which in general is known as heterogeneity, demands efficient methods for effective retrieval of relevant knowledge sources.

In the following sections we will in briefly explain the design choices for an agent based application that enables remote identification of knowledge sources in a distributed environment; one other area of focus is the development of applications that focus on interaction with mobile devices. To this end our application has used special libraries for development of mobile applications. Moreover, in agent based systems performance is a key issue (Burbeck et al, 2004), (Such et al, 2007).

This work focuses on these three research directions: first it presents the principles of an application for knowledge sharing and extraction in a distributed environment using software agents; second, it explains the implementation choices in order this application to be easily accessible using mobile devices. Third it presents performance benchmarks attempting to measure the round trip times for the messages exchanged between the agents and also to measure the efficiency of the system when performing over wireless channels. The structure of the paper is as follows: Section 2 presents in brief related work. Section 3 present the architecture of the system and also gives an example usage scenario. Section 4 presents a system usage scenario; section 5 presents the setup of the experiments and the benchmark results. Section 6 concludes the paper.

2. Related work

In (Such et al, 2007) the authors perform different benchmarks measuring the response times for different scenarios; the authors check the performance of the system on N-hosts. Each benchmark is based on the performance monitoring of a multi-agent system running on different hosts. For different scenarios, the measure result was the average round trip time (RTT) of each message, i.e., the time elapsed from when a sender agent sends a message until it receives the same message sent back by a receiver agent.

In (Vrba, 2003) four different agent platforms were compared in respect to their messaging service; it turned out that JADE provides the most efficient platform compared to FIPA-OS, Jack, and ZEUS. One characteristic of this approach is that the benchmark is adjusted to the comparison of these platforms without internal details to be given; therefore the results cannot be easily extended to other platforms.

(Burbeck et al, 2004) compared the JADE platform with two other platforms. They argue about JADE's superiority due to the fact that it is built upon the Java RMI model. However apart from this claim there is no adequate evidence to support this theoretical claim.

In relevant literature (Such et al, 2007) (Burbeck et al, 2004) the JADE has been established as an extendable platform that performs adequately well in respect to other platforms. In addition it has performed well to relative benchmarks. In our work we try to focus on the performance of the JADE platform and we try to measure its capabilities over wireless environments.

In (Camacho et al, 2002) the authors request several documents with an application consisting of a number of web agents. The authors measure the number of documents requested in the unit of time for different numbers of web agents each time and varying number of documents.

In (Cortese et al, 2002) the authors present the results of scalability and performance measurements of the messaging transport system of the JADE agent platform; they perform different tests on the same and different hosts and show that it performs efficiently in different scenarios for a number of containers up to 1000. Messaging performance for intra-platform configuration highlight that JADE does not introduce relevant overhead compared to its underlining technology Java RMI, and that it well exploits different configurations of the agent platforms

3. System architecture

The system developed implements a distributed knowledge base management system. This system enables retrieval of files (text, images) from distributed locations. The developed system comprises from several modules:

- The distributed document management module: Different files are shared on different network locations. The system enables identification of the files by querying the different network locations; meta-data are stored on each location allowing thus easier querying upon the different file management modules.
- The agent based module; this module is responsible for querying the different sub-modules across the network locations and the authorization module that carries the security credentials. In order to perform authentication related processes each agent carries a SAML compatible message. SAML is an open framework that enables single sign on capabilities through URL based authentication tokens.
- The ontology management module that provides a common framework for the communication between the different agents. In that way each agent is able to interoperate and communicate with others, provided that the different messages are mapped in terms of different ontologies.
- By looking into more detail the internals of the software agent module, it consists of the following sub-modules:
- The Remote Management Agent (RMA), which acts as a graphics console for the management and platform control. JADE preserves connection between multiple RMA's by multicasting several events.
- The Dummy Agent is a tool for detection and correction of errors. It consists of the GUI and an agent. It composes ACL messages and communicates with other agents. It also supports reception and sending of time-stamped messages.
- The Sniffer is an agent that captures ACL messages and through a graphical interface represents them as a series of UML sequence messages.
- The Introspector is an agent that enables the monitoring of the lifecycle of an agent, the exchanged messages between different agents and also enables behavior monitoring of agents.
- The Directory Facilitator (DF) is an agent that provides yellow pages services between different agent domains. It enables control between different knowledge bases, unification between different knowledge-bases through cooperation of different agents and also remote control between parent-child relationships of different domains.
- The LogManagerAgent is an agent that allows time logging for different classes, inherent to the platform as well as custom built that are necessary to the application.
- The SocketProxyAgent is a simple agent that acts as a gateway between the agent management platform and a normal TCP/IP socket connection. ACL messages are transformed into ASCII and accordingly they are being transmitted through a socket connection. This agent is useful in the presence of firewalls and in order to support interactions between the platform and Java applets in a web browser.

Whenever a new container starts to operate an internal RMI registry is created at the current host, which listens to a specific predefined port. Accordingly, it notifies the internal to the system agents (ACC, AMS, and DF). An agent container table is responsible to register and keep updated the list of active agents. In order to increase the system's performance each container stores to its cache a reference to other containers as soon as a new message to them has been sent.

JADE stores appropriate tables of active agents and therefore it selects the most efficient type of message according to the position of the receiving agents. When a new message is sent we have the following options:

- If the receiving agent lives within the same container the message is passed to the receiver by using a message event and without any type of message translation,
- If the receiving agent is JADE-based but lives within a different container, then the ACL message is sent using the RMI communication model.
- If the receiver resides within a different agent platform then a specific protocol (IIOP) and an IDL interface are used according to the FIPA standards; the process includes a translation of the ACL message to string and the execution of a remote call using the IIOP protocol. At the receiver's side the IIOP will produce a Java String and finally it will be transformed to an ACL message.

A simplified picture of the overall system architecture is depicted in Fig. 1. It presents the different knowledge repositories and the pair of agents for each different container on the different hosts that store these repositories. The platform specific agents are not depicted but only the custom made that support the function of the system.

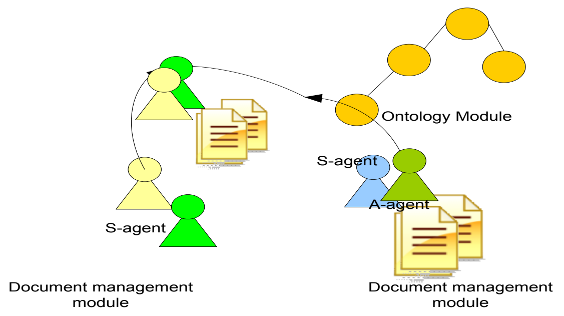

Fig. 1. Overall system architecture

4. System usage scenario

In this section we will briefly analyze the operation of our framework explaining in detail the role of each module. We consider a user who is looking for text files or images that might be relevant to a specific query. The first step involves the user agent that requests authentication permission. If the authentication is successful the authentication agent acquires a token that helps interact with different Search agents. Accordingly the user is able to send a query through an appropriate interface. The search results are sent back to the search agent and if the search is successful a URL with the actual location of the file is returned. The query is also forwarded to the other domains and for each one the search results are displayed and presented to the user through the user agent. The role of the ontology management module is essential in relieving the load of the query in case where the repository does not contain knowledge assets related to the query. For example if a user asks a medical term and the repository does not contain relevant files then the query is forwarded immediately without explicitly querying the files it contains. It is essential at this point to note that in order our application to be easily accessible by mobile devices we have used the JADE LEAP libraries which are designed for lightweight implementations, since mobile devices are resource constrained.

Using the sniffer tool we can see some of the messages exchanged between the agents; this tool enables capturing of messages by means of UML sequence diagram messages (fig. 2)

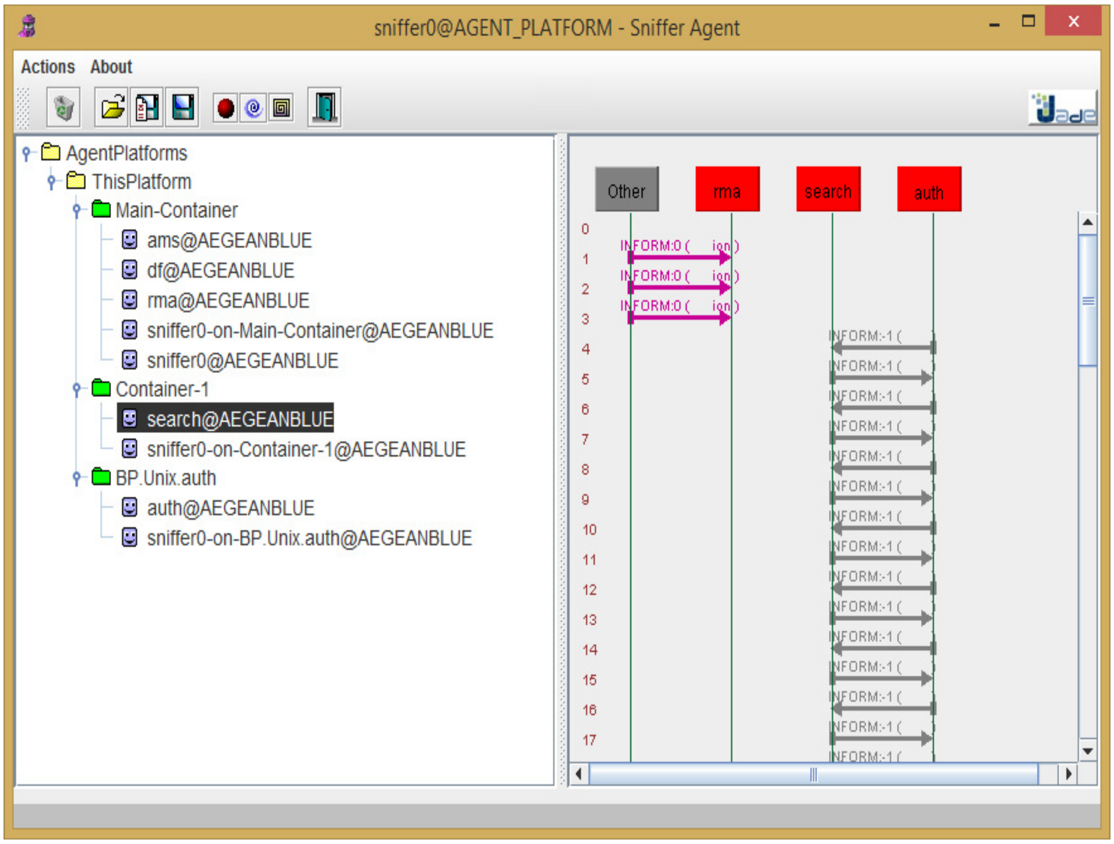

Fig. 2. Communication between the agents

5. Performance benchmarks

In this section we will present the performance benchmarks of our system; we consider the system performing in a wireless scenario, where the users initiate requests using mobile devices. As aforementioned, the JADE LEAP libraries have been used since they demand fewer resources; this is essential when we refer to mobile devices, which in general lack hardware capabilities.

For a benchmark over a wireless channel different parameters can be measured; one such can be the time elapsed from the moment one query is initiated until the answer is sent back. This parameter can be affected either by the nature of the wireless environment (protocols used, number of messages exchanged, jitter etc) but also by the nature of the agent application and the response times of the used platform. We consider for start an 802.11 environment with a capacity of 1Mbps. We will calculate the total delays, which comprise of the average times required for transmission of the message and the times required by the platform to generate and send the message. This time is calculated by measuring the time provided by the system clock. We consider a message length of 800bits (with the network protocol overhead included). We consider that the messages sent follow a Poisson distribution with a mean value for packet generation equal to 10 messages/sec. The simulation was performed for a total of 300sec. We used for the wireless channel simulation the Pamvotis simulator (Pamvotis, 2013). We perform different simulation scenarios for 10, 20, 30 and 40 hosts (the number of agents surpasses the number of hosts).

In the experiments we consider distinct cases (Figure 3): i) One container ii) Two containers on same host and iii) Two containers on different hosts. In the first scenario we measure the total delays for a message to arrive from one end to the other. We additionally measure the times required by the agent platform and finally we calculate the total time required (by both the wireless channel and the agent platform). The results are depicted in Figure 3.

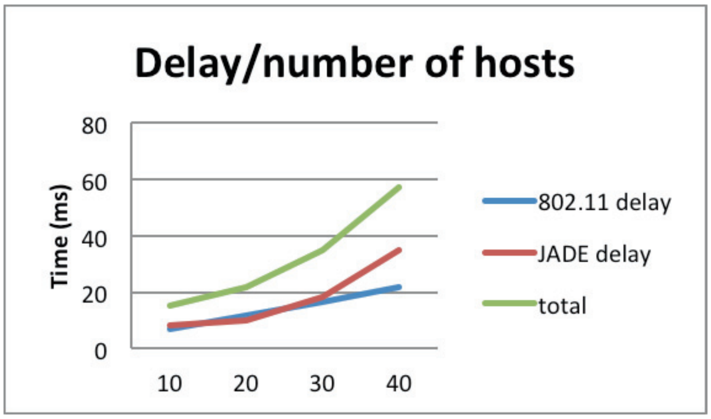

Fig. 3. Message round trip total delays in respect to number of hosts

In the second experiment we measure the round trip times in respect to the number of agents; we consider two cases, the first being communication between agents in the same host and the second when two containers reside in two different hosts. In the second case we see the times needed being slightly lower, which is due to the lower CPU processing costs when communication is split between two different processors (Figure 4).

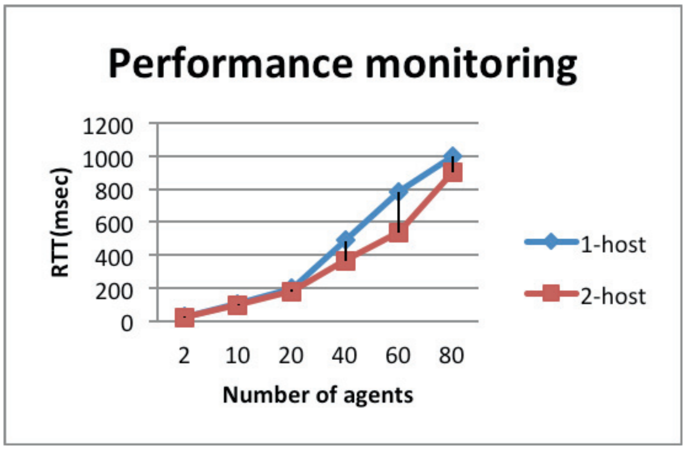

Fig. 4. Round trip times for the platform in respect to number of agents

Overall we see that for an average number of agents both the total delays as well as the performance of the system are quite acceptable.

6. Conclusions

Software agents have been lately integrated to a large number of applications. With the rise of applications for mobile devices it is necessary to explore also the possibilities of software agent applications over wireless devices. In this paper we describe an application that enables resource sharing in a distributed environment using software agents. We describe the implementation details and also provide a testbed that enables benchmarking and performance monitoring.

In our experiments we saw that the total times required for agent communication including the delays inherent to the platform and the times required due to the 802.11 nature, are very reasonable. We also measured the performance in comparison to the number of agents and found that there is a linear correlation of the number of agents in respect to the response times.

In overall, the JADE software agents platform has been proved a quite reliable platform for the development of applications for both wired or wireless infrastructures. In the future we are planning scenarios for remote monitoring of large number of hosts and we also plan to perform measurements with larger number of hosts.

Acknowledgements

This research has been co-funded by the European Union (Social Fund) and Greek national resources under the framework of the "Archimedes III: Funding of Research Groups in TEI of Athens" project of the Education and Lifelong Learning operational Programme.

References

Bellifemine, F., Caire, G., Poggi, A., Rimassa G.: JADE: A software framework for developing multi-agent applications. Lessons learned. Information & Software Technology 50(1-2): 10-21 (2008).

Belsis P., Vassis D., Skourlas C., Pantziou G., “Secure dissemination of electronic healthcare records in distributed wireless environments”, Proceedings of 21st International Congress of the European Federation for Medical Informatics (S. Anderssen et al. eds), Goteborg, pp. 661-666, Sweden May 2008, IOS Press

Burbeck, K., Garpe, D., Nadjm-Tehrani, S.: Scale-up and performance studies of three agent platforms. In: IPCCC 2004 (2004)

Camacho, D., Aler, R., Castro, C., Molina, J.M.: Performance evaluation of zeus, jade, and skeletonagent frameworks. In: IEEE International Conference on Systems, Man and Cybernetics, 2002 (2002).

E. Cortese, F. Quarta, G. Vitaglione, “Scalability and Performance of JADE Message Transport System”, AAMAS Workshop on Agenticies, Bologna, July 2002.

Pamvotis network simulator, www.pamvotis.org (accessed 2013).

Such, J.M., Alberola, J.M., Mulet, L., Espinosa, A., Garcia-Fornes, A., Botti, V.: Large-scale multiagent platform benchmarks. In:anguages, methodologies and Development tools for multi-agent systems (LADS 2007). Proceedings of the Multi-Agent Logics, Languages, and Organisations - Federated Workshops. pp. 192–204 (2007).

Turner, P.J., Jennings, N.R “Improving Scalability of Multi-Agent Systems”, Proc.1st Int’l Workshop Infrastructure for Scalable Multi-Agent Systems. June 2000.

Vrba, P.: Java-based agent platform evaluation. In: Proceedings of the HoloMAS 2003. pp. 47–58 (2003).

Zafeiris V., Doulkeridis, C., Belsis P., I. Chalaris “Agent-mediated Knowledge Management in Multiple Autonomous Domains, International Conference on Autonomous Agents and Multiagent Systems-Workshop on Agent Mediated Knowledge Management, Univ. of Ulterecht Nederlands, July 2005, pp. 97-112.