Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2008, Article ID 470856, 7 pages
doi:10.1155/2008/470856

Research Article

Messaging Performance of FIPA Interaction Protocols in Networked Embedded Controllers

Omar Jehovani López Orozco,[1] Jose Luís Martínez Lastra,[1] José A. Pérez García,[2] and María de los Ángeles Cavia Soto[3]

[1] *Institute of Production Engineering, Tampere University of Technology, Korkeakoulunkatu 6, 33101 Tampere, Finland*
[2] *E.T.S. de Ingenieria Industrial, Univerisdad de Vigo, 36310 Vigo, Spain*
[3] *Departamento de Ingenieria Electrica y Energetica, Universidad de Cantabria, 39005 Santander, Spain*

Correspondence should be addressed to Jose Luís Martínez Lastra, lastra@ieee.org

Received 1 February 2007; Revised 18 June 2007; Accepted 5 November 2007

Recommended by Valeriy Vyatkin

Agent-based technologies in production control systems could facilitate seamless reconfiguration and integration of mechatronic devices/modules into systems. Advances in embedded controllers which are continuously improving computational capabilities allow for software modularization and distribution of decisions. Agent platforms running on embedded controllers could hide the complexity of bootstrap and communication. Therefore, it is important to investigate the messaging performance of the agents whose main motivation is the resource allocation in manufacturing systems (i.e., conveyor system). The tests were implemented using the FIPA-compliant JADE-LEAP agent platform. Agent containers were distributed through networked embedded controllers, and agents were communicating using request and contract-net FIPA interaction protocols. The test scenarios are organized in intercontainer and intracontainer communications. The work shows the messaging performance for the different test scenarios using both interaction protocols.

1. INTRODUCTION

Rapid reconfiguration of manufacturing systems in response to continuous changing conditions is one of the challenges identified by the survey on visionary manufacturing designed to forecast manufacturing challenges in 2020 [1]. The survey identified strategic technology areas such as (a) adaptable integrated equipment, process, and systems that can be readily reconfigured, (b) technologies to convert information into knowledge for effective decision making, and (c) software for intelligent collaboration systems.

Agent paradigm is one of the main players in the development and/or merging of these strategic technologies. An agent in the context of this work is a software entity which has information about its own environment and that is able to request information from other agents. It is also capable of independent or collaborative actions. The use of agent technologies and multiagent systems through many fields in production systems has proved their value in different applications through different fields [2].

Decision mechanisms implemented with agents and control algorithms can be distributed through networked embedded controllers in production lines as suggested in [2, 3]. Therefore, agents would need an organized and standardized way to interact. The Foundation for Intelligent Physical Agents (FIPA) has developed guidelines for the development of agent platforms which include an abstract architecture and a set of interaction protocols (IPs) [4].

Key points in the development of multiagent systems are the design of behaviors, the bootstrap, and the interactions. The IPs used in this work are of request and contract-net types are shown in Figures 1 and 2, respectively. The request protocol can be used for peer-to-peer interactions with other agents, that is, to obtain information or to request the execution of a task to other agents. The contract-net protocol is used for decentralized task allocations; it is a distributed negotiation model based on the notion of call for bids on markets [5]. The novelty of the work stems from the effective distribution of negotiating peers through devices which have very small footprint.

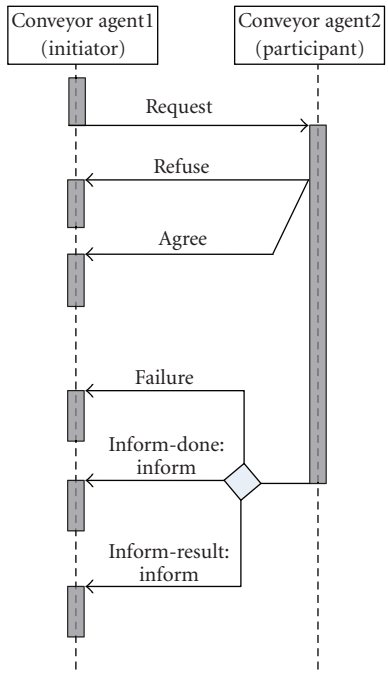

Figure 1: FIPA request interaction protocol [6].

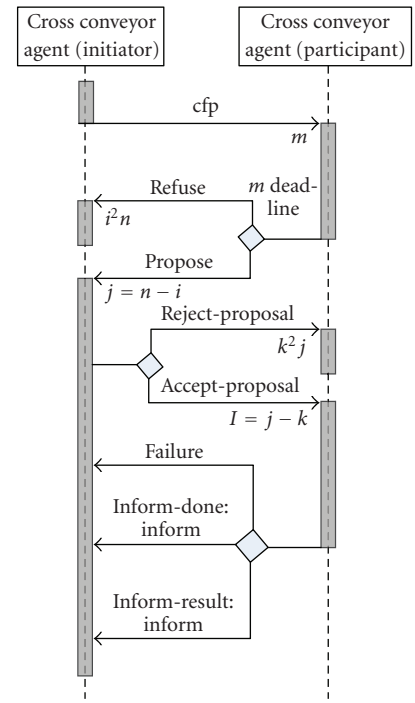

Figure 2: FIPA contract-net interaction protocol [7].

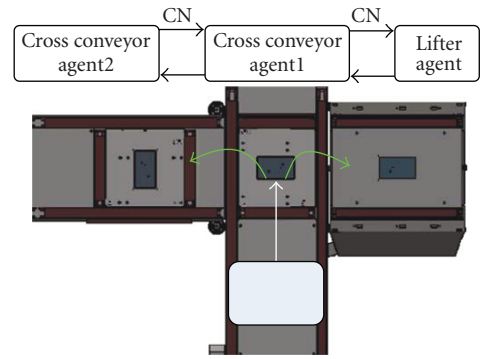

Figure 3: A cross-conveyor is a shared resource in which an agent could coordinate by means of the contract-net protocol to which the attached conveyors will send a pallet.

The main motivation to study the messaging performance is given by the limitations of the embedded devices. Even though the interaction protocols represent an organized and meaningful way to interact, they impose more constraints on the timely flow of information, and consequently on the execution time of tasks in manufacturing systems. Therefore, the study of the messaging time performance under different test scenarios provides information about the feasibility to coordinate agents which are distributed within the embedded controllers.

This work is organized as follows Section 2 includes a description of related work. Section 3 explains the FIPA interaction protocols utilized in the messaging performance. Section 4 provides a short description of the JADE-LEAP and the pointe controller (the software and hardware platforms). Section 5 describes the test scenarios and shows the results of the messaging performance. A short discussion is presented in Section 6, and finally the conclusions are in Section 7.

2. RELATED WORK

Traditional centralized control systems are inadequate for complex production facilities exposed to continuous changes. It has been stated in [3, 5] that distribution of control algorithms and decision mechanisms could bring benefits such as system scalability and robustness. Agents could be distributed providing logical intelligence through production lines. There have been agent-based control applications for different industrial environments. These are distributed solutions applied in areas such as real-time manufacturing control, complex operations management (planning, scheduling, bootstrap, monitoring), and coordinating virtual enterprises [2]. Simulation testbeds as in [8], industrial applications like in [9–11], or the rod steel production and navy chilled water system prototype [12] have been proved to be the benefits and possibilities of agent technology.

There are some works which have studied the scalability and performance of agent platforms and agent-based applications, respectively. A benchmark on message transport system is documented in [13]. The agent platform used is JADE, and it was running on personal computers, with microprocessors at 800 MHz and 256 MB of RAM, and connected via 100 Mbps Ethernet LAN. The general scenarios are the intercontainer and intracontainer messaging for different quantities of agent pairs. The round trip time was similar in

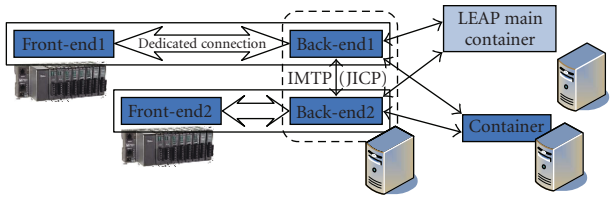

Figure 4: JADE-LEAP front end and back end containers.

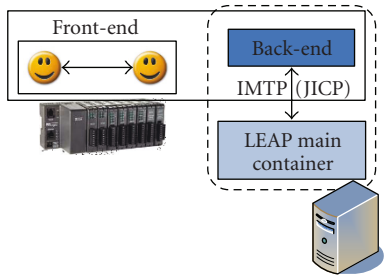

Figure 5: First scenario with the request interaction protocol; initiator and participant are running in the same front-end container.

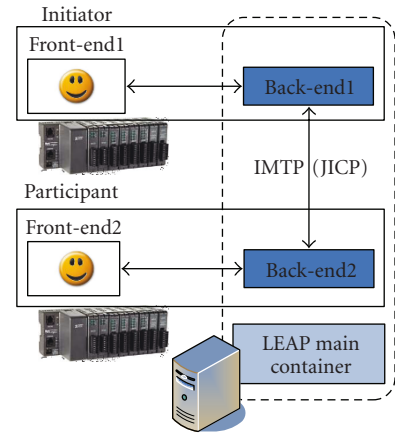

Figure 6: Second scenario for the request interaction protocol; agent initiator and agent participant are running in different front-end containers.

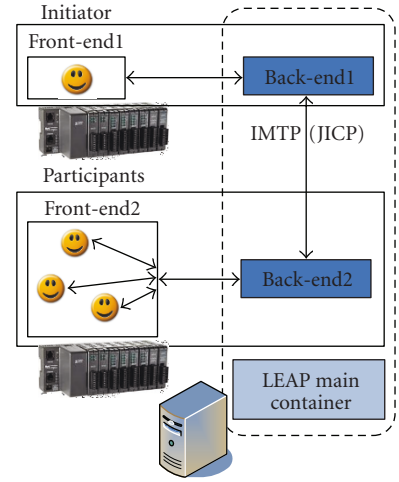

Figure 7: Second scenario for testing the contract-net interaction protocol; initiator and participant agents are hosted in different front-end containers.

the intra- and intercontainer; it oscillates around 12 milliseconds.

Vrba presented a message sending speed benchmark of Java-based agent platforms [14]; the messages were single round tripe and the hardware running the different platforms was a PC. It is shown in the benchmark that the fastest platform in delivering messages was JADE. In addition, the Java files of the agents were the smallest among the other tested platforms. Vrba and Hrdonka in [15] proposes an approach to solve problems related to material handling systems using multiagent systems and ontologies. The simulation developed to prove the concept was implemented with JADE.

Laukkanen et al. in [16] investigated the performance of the agent platform MicroFIPA-OS running on handled devices and wireless transmission media at 9600 bps. The work shows the timing response of the request interaction protocol and the performance of searches in the directory facilitator (DF); the simple request takes around 861 (±484) milliseconds and a search could take 7765 (±568) milliseconds. According to [3], it is expected that agents' response times (in the high-level control layer) oscillate between 100 milliseconds and 10 seconds.

A previous work of the authors [17] measured the average round trip time (aRTT) of single inform messages. The purpose of the test was to observe the scalability in terms of pairs of agents. The agent platform used was LEAP running in split configuration on embedded controllers. The test scenarios included were intracontainer and intercontainer messaging. For the first case, the aRTT obtained was around 8 milliseconds (5 pairs of agents), and for the second case, the time for one pair of agents was 465 milliseconds.

3. FIPA INTERACTION PROTOCOLS

Agent communication language (ACL) is a declarative language; it is sufficiently expressive to enable communication of all sorts of information (i.e., states, definitions, assumptions, rules, data). ACL is a language with well-defined semantics, syntax, and pragmatics; its two main components are the communicative act and the content of the message. Agents can use an ACL in order to encode the messages [7]. The ACL is composed from the interaction protocols (IPs) standardized by the Foundation for Interoperability of Physical Agents (FIPA).

The collection of FIPA standards promotes interoperability of heterogeneous agent platforms; it provides a set of specifications and an abstract architecture. The former includes different interaction protocols for agent communication; the latter is a reference architecture to promote interoperability

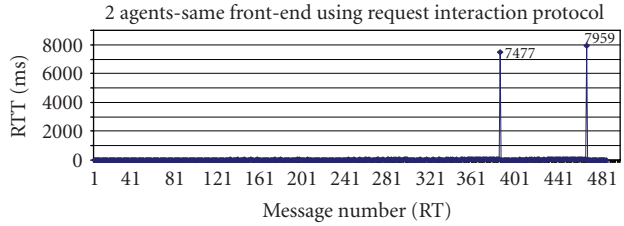

Figure 8: FIPA request interaction protocol; initiator and participant are in the same PTC.

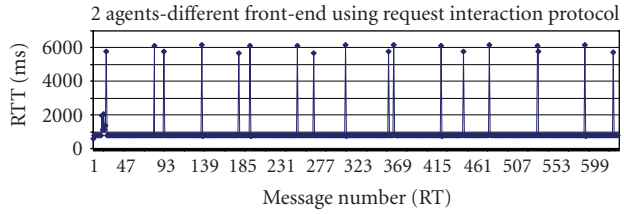

Figure 9: Performance of request interaction protocol; initiator and participant are in different controllers.

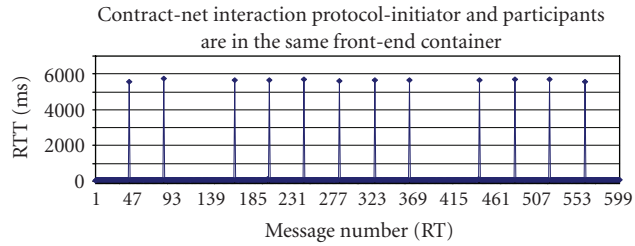

Figure 10: Performance of contract-net interaction protocol; agent initiator and agent participants are running in the same controller.

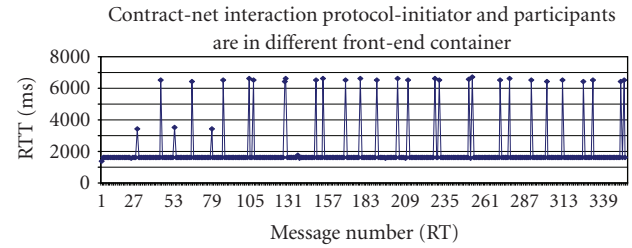

Figure 11: Messaging performance of the contract-net interaction protocol; initiator and participants are in different controllers.

among different platforms. In peer-to-peer agent interactions, protocols such as request, query, and contract-net protocols provide an organized and meaningful way to exchange information via the contents of the messages. IPs favor coordination and promote interoperability among agents.

3.1. Contract-net interaction protocol

In contract-net interaction protocol, the relation between agent clients (managers) and agent suppliers (bidders) is created in a call for bids and an evaluation of the proposal submitted by the bidders to the managers [18]. Thus, an agent requests that other agents submit their proposals with appropriate bids to the task. The manager agent reviews the bids and chooses the bidder with the best proposal. The negotiation process is carried out in four steps (also shown in Figure 2).

(a) The agent acting as manager sends calls for bids to the agents which would be able to perform certain task(s).
(b) The bidders use the description of the task to build a proposal that they will send to the manager.
(c) The manager receives and evaluates the proposal and assigns the task to the best bidder.
(d) During the last step, the bidder to which the task is assigned sends the manager a message to confirm its intention to do the request task.

LEAP provides API to directly use the FIPA IPs such as request and contract-net.

Figure 3 shows a scenario in which the contract-net protocol coordinates the interaction between a cross-conveyor and three attached conveyors. The central conveyor has an agent which is the manager or the initiator; it sends a CFP message in order to know the availability of the other conveyors, by means of the proposals. The central conveyor could decide which proposal best suits its own goals.

4. MULTIAGENT PLATFORM FOR NETWORKED EMBEDDED DEVICES

The use of networked embedded controllers to distribute decision mechanisms by means of software agents improves qualitative attributes such as scalability and reconfigurability of modular systems. Agents running on networked embedded devices could deal better with distributed problems (i.e., resource allocation). Agent platforms provide basic services for administrating the life cycle of agents, the communication media, and some other services.

4.1. Networked embedded controllers

Advantages of networked embedded controllers compared to traditional programmable logic controllers are the enhanced connectivity infrastructure, communication protocols, and new programming paradigms. The need to simplify the point-to-point wiring connections in order to facilitate changes and maintenance is just one of the reasons to use distributed systems.

C, C++, and Java are the most popular and widespread languages for programming industrial networked embedded devices. Traditionally, C code has been more efficient than the code generated by the other two languages. Nevertheless, state-of-the-art controllers which directly execute Java virtual machine byte code had shown similar execution efficiency.

The garbage collector (GC) of Java is one of the main functional differences compared to C and C++. The GC is responsible for freeing memory of Java processors by erasing

unnecessary objects, as opposed to C and C++ for which the programmers have to explicitly take care of the memory. The GC in devices such as personal computers could pass unnoticed in most of the cases, but for embedded devices it could have serious effects on the performance. In order to minimize the effects, direct Java CPUs such as the microprocessor AJ-100 [19] have been designed with dual heaps; one of the heaps is meant for time critical operations and the other heap for noncritical operations. The heap which does not generate garbage ensures that processes such as gathering I/Os, real-time control, and communications are not interrupted. Applications running on the heap which generates garbage are preempted during the GC execution.

The pointe controller or PTC5800 is an embedded controller that offers more functionality than traditional programmable logic controllers (PLCs). The processing unit is an aJile microprocessor at 100 MHz; it directly processes native Java virtual machine (JVM) byte code. The PTC uses the connected limited device configuration (CLDC) which outlines the most basic set of libraries and virtual machine features that must be present in each implementation of Java 2 Microedition (J2ME) environment on resource-constrained devices. The connectivity features of these controllers are Modbus TCP/IP, TCP/IP Ethernet, and a serial port.

4.2. The multiagent platform

An agent platform is a middleware which provides a set of normative and optional services for the deployment and execution of peer-to-peer applications. FIPA has proposed a reference architecture and interaction protocols to facilitate interoperability among different agent platforms. The set of normative services includes a life cycle management, white and yellow pages, and message transport service.

There are many agent platforms available (the reader interested in a detailed list should refer to [14]). The Java Agent DEvelopment Framework (JADE) is a middleware fully compliant with the FIPA specifications. The elements that JADE includes from the FIPA abstract architecture are an agent directory, a directory service, a message transport, and agent communication language.

The standard version of JADE is too big to be executed by embedded devices with small footprint. Nevertheless, there is a light version of JADE called light-weight and extensible agent platform (LEAP) which can be used for devices with limited capabilities.

4.3. The JADE-LEAP platform

The LEAP platform is a light version of JADE; it is meant for devices with limited capabilities which use either CLDC or the connected device configuration (CDC) version of Java microedition (J2ME). The LEAP run-time environment must be active on the device before agents can be executed. Each instance of the LEAP run-time is called container and a group of containers composes a platform. A container provides basic functionality for agents hosted within it. The deployment of multiagent systems on embedded controllers such as the pointe controller requires a container running on the embedded controller. LEAP was not thought for any device in particular; instead, it was developed as a framework. The LEAP's API can be modified to fulfill different families of devices. LEAP can be deployed according to a set of profiles identifying the functionality available on each particular device. A MIDlet has to be created from the LEAP source code in order to deploy agents on the pointe controller.

LEAP's run-time environment could be implemented either as a stand-alone or a split configuration. In the stand-alone execution mode, the entire container's functionality is executed on the target device. On the other hand, in the split execution mode, a container is separated into a front end (running on the embedded device) and a back end (running on a host with J2SE). The front end and back end maintain communication through a permanent connection. The split execution mode is better for constrained devices since the front end is more lightweight than a complete container, the bootstrap phase is faster, and less amount of bytes is transmitted over the network.

The agent management system (AMS) and the directory facilitator (DF) are provided by the platform's main container. The front end is able to detect a disconnection with the back end. Thus, if that happens, it keeps trying to connect again for a period of time. The back end is a dispatcher for messages sent by other containers within the same platform whose destinations are agents on the front end [20]. LEAP internal message transport protocol (IMTP) is based on a proprietary protocol called JICP (JADE intercontainer protocol) [6]. The LEAP IMTP has a command dispatcher which is responsible for serializing and deserializing JADE horizontal commands, assigning the proper intercontainer protocol (ICP) to serialized commands, and routing commands received from the ICPs to the local container. The selection of a communication depends on the location of the sender and receiver agents.

(1) Communication between agents hosted on different containers which belong to the same platform uses IMTP (see Figure 4).

(2) If agents are in the same container, the message is sent using event passing. Naturally, the message is not serialized, but cloned, and the object reference is passed to the agent receiver [13].

The main features of the embedded controllers used during the tests are shown in Table 1.

Table 1: Main features of the embedded controllers used in the experiments.

Device components	Characteristics
Embedded controller	PTC-5800
Microprocessor	AJ-100 at 100 MHz
RAM	2 MB
Flash	4 MB
Java virtual machine	Profile: CLDC 1.0.3, J2ME
Connectivity	Ethernet TCP/IP (10 BaseT)

5. MESSAGING PERFORMANCE

The goal of these tests is to study the messaging performance during the agent communication. Request and contract-net are the IPs used in the different test scenarios. Two scenarios are presented regarding the allocation of the sender and receiver agents. The first scenario is the intracontainer communication; in this case, agents are running on the same front-end container. The second scenario is the intercontainer communication, and agents are distributed in two front-end containers.

The physical message passing among the logical controllers is based on the TCP/IP protocol. The PTCs were connected through an IEEE 802.3/Ethernet 10 Base-T multiport repeater. The length of each FIPA message is around 282 characters. This value can change; it depends on the content of the message. For the purpose of the tests, only the agents' threads were running within soft real-time heap of the AJ-100 microprocessor. In addition, the length of the messages was held constant. In the case of the request protocol, the amount of sent and received messages was 500, and in the contract-net case, the amount was 180 messages.

5.1. Messaging performance of the request interaction protocol

The first scenario is illustrated in Figure 5. There are two agents, initiator, and participant which are running in the same front end.

Figure 8 shows the time performance of the request interaction protocol with continuous sending of messages in the same front-end container. Without considering the garbage collection, the highest peaks are below 35 milliseconds. There are two peaks which exceed 7000 milliseconds; they are the consequence of rescheduled messages due to garbage collection procedures.

The second test was set up using two controllers, the agent initiator was running in one controller, and the agent participant on the other (see Figure 6). Figure 9 shows the results of the experiments. It can be seen without considering the peaks that it takes around 790 milliseconds to complete the interaction protocol.

5.2. Messaging performance of contract-net interaction protocol

The first test scenario with contract-net protocol was performed with one agent initiator and three agent participants; all were running on the same controller. The average time interval for the round trip time (RTT) was 80–130 milliseconds (see Figure 10). Nevertheless, the peaks due to the garbage collection procedure are within the interval of 5700–6000 milliseconds.

In the second test, two controllers were used, one with an initiator agent and the other with 3 participants (see Figure 7). The results are shown in Figure 11. The average RTT is around 1610 milliseconds.

Table 2: Transferring times for a cross-conveyor to its adjacent conveyors, and between two single conveyors.

Origin	Destination	Time (ms)
Center cross	Right cross	3820
Center cross	Left cross	3820
Center cross	Middle conveyor	2500
Left cross	Left conveyor	3730

6. DISCUSSION

The garbage collection in PTC-5800 is scheduled automatically when it is needed; it could happen every 10 minutes or 20 seconds or at any time. When it occurs, it preempts noncritical threads for a few seconds. Two kinds of threads are classified according to their priority in critical threads and noncritical threads within the PTC. Agents within a front-end container, hosted in the PTC, are scheduled as noncritical threads; this means that they could be affected by the execution of the garbage collection thread.

The behavior of the two agents within the same container and for both interaction protocols shows how the RTT increases in intervals of 70–80 RTTs. This is due to the fact that every sent message is received on the private queue of the receiver agent. Therefore, this together with the creation of message objects (which is the most demanding processing protocol) deteriorates the messaging performance until the activation of GC.

The RTT of agents in different containers is naturally bigger than that of agents within the same container; the ACL messages are marshaled prior to sending and unmarshaled at the destination. The most demanding IP was the contract-net for obvious reasons; it has to deal with more messages, and the interaction implies analysis of the received bids. Nevertheless, the messaging performance measurements have demonstrated that even for the most time-demanding interaction, the RTT is still below the maximum limit stated in the literature and the transferring times shown in Table 2.

The results presented in this work are proportionally similar (considering the number of messages and the number of agent pairs) compared to the results obtained in a previous work [17].

Table 2 shows the transferring times for two types of conveyors, the cross-conveyor like the one shown in Figure 3, and the single conveyors. According to these transferring times, agents would have time to negotiate in advance the next action to be taken. This assumption still will need some experimentation to be proved.

7. CONCLUSION

This work has shown the messaging performance of two FIPA interaction protocols: request and contract-net. This work has been conducted through different test scenarios using a Java-based controller and an adapted JADE-LEAP platform. The results showed the round trip time of the entire interaction protocols; this is from the beginning of the

interaction ending with the inform-done message. The messaging performance is affected by the garbage collection which preempts the execution of any other thread.

The transferring times to move a pallet from a central conveyor to one of the attached conveyors could allow coordinating/negotiating in advance the next movement. This will be part of future work.

ACKNOWLEDGMENTS

This work was supported by the National Technology Agency of Finland (TEKES), and the participant companies of the IMPRONTA applied research project: Nokia Mobile Phones, Perlos, FlexLink Automation, Cencorp, and Photonium.

REFERENCES

[1] The National Academy of Sciences, Visionary Manufacturing Challenges for 2020, December 2006, http://newton.nap.edu/html/visionary/summary.html.

[2] V. Maríık and D. McFarlane, "Industrial adoption of agent-based technologies," *Intelligent Systems*, vol. 20, no. 1, pp. 27–35, 2005.

[3] J. Christensen, "HMS/FB architecture and its implementation," in *Agent-Based Manufacturing: Advances in the Holonic Approach*, pp. 53–87, Springer, Berlin, Germany, 2003.

[4] FIPA TC Architecture. FIPASC00001L. FIPA Abstract Architecture Specification. Standard, Geneva, Switzerland, 2002.

[5] R. G. Smith, "The contract net protocol: high-level communication and control in a distributed problem solver," *Transactions on Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.

[6] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*, Wiley Series in Agent Technology, John Wiley & Sons, New York, NY, USA, 2007.

[7] P. Charlton, R. Cattoni, A. Potrich, and E. Mamdani, "Evaluating the FIPA standards and its role in achieving cooperation in multi-agent systems," in *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, p. 230, Island of Maui, Hawaii, USA, January 2000.

[8] W. Brennan and O. William, "A simulation test-bed to evaluate multi-agent control of manufactuirng systems," in *Proceedings of the 37th Winter Simulation Conference (WSC '00)*, vol. 2, pp. 1747–1756, Orlando, Fla, USA, December 2000.

[9] P. Tichý, P. Šlechta, F. P. Maturana, and S. Balasubramanian, "Industrial MAS for planning and control," in *Proceedings of the 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001 on Multi-Agent-Systems and Applications II*, vol. 2322 of *Lecture Notes in Computer Science*, pp. 280–295, Springer, Prague, Czech Republic, July 2002.

[10] R. Schoop, A. W. Colombo, B. Suessman, and R. Neubert, "Industrial experiences, trends and future requirements on agent-based intelligent automation," in *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society (IECON '02)*, vol. 4, pp. 2978–2983, Sevilla, Spain, November 2002.

[11] S. Bussmann and K. Schild, "Self-organizing manufacturing control: an industrial application of agent technology," in *Proceedings of the 4th International Conference on MultiAgent Systems (ICMAS '00)*, pp. 87–94, Boston, Mass, USA, July 2000.

[12] K. Hall, R. Staron, and P. Vrba, "Holonic and agent-based control," in *Proceedings of the 16th International Federation of Automatic Control World Congress (IFAC '05)*, Prague, Czech Republic, July 2005.

[13] E. Cortese, F. Quarta, and G. Vitaglione, "Scalability and performance of JADE message transport system," in *Proceedings of the AAMAS Workshop on AgentCities*, pp. 1–6, Bologna, Italy, July 2002.

[14] P. Vrba, "JAVA-based agent platform evaluation," in *Proceedings of the 1st International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS '03)*, vol. 2744 of *Lecture Notes in Computer Science*, pp. 47–58, Springer, Prague, Czech Republic, September 2003.

[15] P. Vrba and V. Hrdonka, "Material handling problem: FIPA compliant agent implementation," in *Proceedings of the 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001 on Multi-Agent-Systems and Applications II*, vol. 2322 of *Lecture Notes in Computer Science*, pp. 268–279, Springer, Prague, Czech Republic, 2002.

[16] M. Laukkanen, S. Tarkoma, and J. Leinonen, "FIPA-OS agent platform for small-footprint devices," in *Proceedings of the 8th International Workshop on Intelligent Agents VIII*, J.-J. C. Meyer and M. Tambe, Eds., vol. 2333 of *Lecture Notes in Computer Science*, pp. 447–460, Springer, Seattle, Wash, USA, August 2002.

[17] O. López and J. Martinez-Lastra, "Performance measurement of a multiagent system," in *Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM '06)*, Saint-Etienne, France, May 2006.

[18] SC00029H, "FIPA Contract Net Interaction Protocol Specification," FIPA TC Communication, December 2002.

[19] aJile Microprocessor, February 2007, http://www.jempower.com/ajile/content/view/21/28/.

[20] F. Bergenti and A. Poggi,, *LEAP: A FIPA Platform for Handheld and Mobile Devices*, Springer, Berlin, Germany, 2002.