Monitoring Agent Communication in Soft Real-time Environments

Andreas L. Symeonidis and Pericles A. Mitkas
Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, GR54124, Thessaloniki, Greece
Informatics and Telematics Institute – CERTH, GR57001, Thessaloniki, Greece
asymeon@eng.auth.gr, mitkas@eng.auth.gr

Abstract

*Real-time systems can be defined as systems operating under specific timing constraints, either hard or soft ones. In principle, agent systems are considered inappropriate for such kinds of systems, due to the asynchronous nature of their communication protocols, which directly influences their temporal behavior. Nevertheless, multi-agent systems could be successfully employed for solving problems where failure to meet a deadline does not have serious consequences, given the existence of a fail-safe system mechanism. Current work focuses on the analysis of multi-agent systems behavior under **such soft real-time constraints**. To this end, **ERMIS** has been developed: an integrated framework that provides the agent developer with the ability to benchmark his/her own architecture and identify its limitations and its optimal timing behavior, under specific hardware/software constraints. A variety of MAS configurations have been tested and indicative results are discussed within the context of this paper.*

1. Introduction

During the last decade software agents have been considered as state-of-the-art in a wide range of applications, where distribution, delegation and diversity are key factors. Agents' ability to function continuously, perceiving the environment through sensors and acting on it through effectors [1] has established a new software paradigm that has enjoyed popularity and acceptability. Various agent system architectures have been proposed, focusing on the different aspects of multi-agent systems (MAS), namely coordination, learning, and adaptation. Nevertheless, communication remains the most important aspect in MAS. Established communication protocols are nowadays implemented as asynchronous message exchange mechanisms, employing one of the existing Agent Communication Languages (ACL). It is *this* asynchronous nature of agent communication that makes agent technology "inappropriate" for real-time applications.

A specific subcategory of such applications, though, can successfully employ agents and enjoy their advantages. *Soft real-time* systems (SRTS) consider timing critical in the sense that specific goals have to be satisfied (and messages exchanged) within a particular time-frame that is small, but not absolutely strict. One could consider monitoring systems as being representative examples of STRS.

Current work focuses on the investigation of issues related to communication timing and discusses a framework built for controlling custom-made MAS timeliness and for identifying their optimal communication threshold. This way, message routing and action execution within pre-specified time intervals are ensured, given the SRTS requirements.

The rest of the paper is organized as follows: Section 2 discusses related work, while Section 3 provides an overview of the developed software framework. Section 4 qualitatively discusses a number of experimental scenarios in order to identify key points in MAS timing, while Section 5 comments on the results, discusses future work and concludes the paper.

2. Related Work

In related bibliography, one may identify but a few efforts that employ agent systems for solving real-time problems. The primary reason for this is that most agent development frameworks are written in Java, where each agent executes its own processing thread, running in parallel with all the others. This is turn generates a problem in *MAS synchronization*, i.e. the state where the respective message exchange monitoring mechanism is able to capture a message cycle (from the sender to the receiver and acknowledgement of receipt), before the initiation of a new message cycle. Specific approaches attempt to solve the problem either theoretically or practically and, according to the approach followed, they can be classified as:

a) Systems employing a mechanism that assists agents in performing their tasks. The term *mechanism* is defined as the way agents communicate, the role each agent uptakes, their autonomy degree etc. Examples of such systems are the TTA (Time-Triggered Architecture) [2], CIRCA (Cooperative Intelligent Real-Time Control Architecture) [3], SIMBA (SIstema Multiagente Basado en ARTIS) architecture [4], AMSIA [5], and LCT [6]. It should be pointed out that this category is the most popular one.

b) Systems that extend a specific agent communication language, in order to allow for the handling of real-time applications. In this direction, one may point out work by DiPippo et. al.[7], proposing a control mechanism along with the extension of the KQML ACL, as well as work by Nair [8], proposing the extension of the KQML for dealing with issues imposed by real-time applications.

c) Finally, a number of experimental analyses exist, dealing with the performance and adjustability of the message monitoring mechanisms of well-known agent development frameworks under various operation conditions. Specifically, Jurasovic et. al. [9] compare the message monitoring mechanisms of Grasshopper and JADE. The authors perform a number of experiments, with varying message sizes and encoding techniques and demonstrate JADE superiority. On the other hand, Chmiel et. al. [10] focus on JADE and attempt to benchmark its performance, with respect to message exchange and agent migration capabilities. Finally, Vitaglione et. al. [11] perform a variety of tests on the adjustability of the JADE messaging service, for agent systems operating in distributed locations.

Extending previous work, the *ERMIS* framework has been developed and is presented. *ERMIS* provides an easy and re-configurable way for setting up and benchmarking new and existing agent systems, while it provides a set of useful statistics for the developer to evaluate his/her system real-life performance. The basic functionality and architecture of *ERMIS* are discussed next.

3. The ERMIS Framework

Through *ERMIS* developers are able to benchmark their system and identify the *synchronization threshold*, i.e. the time interval threshold (in *ms*) above which the system does not have any communication failures (delays in message transition). It provides a multivariate console, in order to define the (groups of) agents to monitor and their messaging priority, and test them against different time intervals and thresholds. The results are presented in aggregate graphs and tables, in an easy to comprehend and compare manner.

3.1. ERMIS core entities

The basic entities implemented within the context of this work are:

- Agent ***MySniffer***: an extension of the JADE *Sniffer,* which monitors message exchange and performs timing control.
- The ***ERMISenderBehavior()***: a ticker behavior assigned to any agent that sends messages (only sending messages).
- The ***ERMISReceiverBehavior()***:a ticker behavior assigned to any agent that receives messages (only receiving messages).
- The ***ERMISenderReceiverBehavior()***: a ticker behavior assigned to any agent that sends and receives messages.

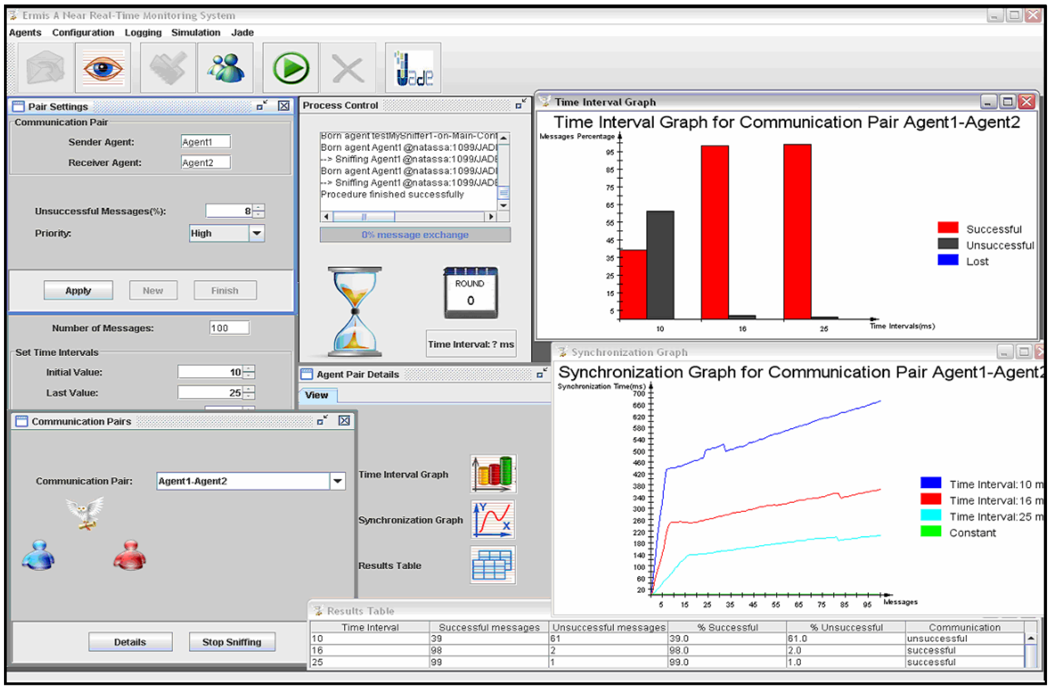
3.2. ERMIS Graphical User Interface

The ERMIS GUI provides user-friendly options for importing the parameters of the MAS to be tested, as well as for viewing and exporting the results of the experiments. The main option panes are:

- ***Common Parameters Option Pane.*** Through this Pane the user defines the number of messages that each communication pair shall exchange, the range of values (in *ms*) for the system to be tested against and number of intervals (through the interval definition function).
- ***Communication Settings Option Pane.*** Through this pane the user defines all the Communication Pair-related settings: the agents to be involved, the Unsuccessful Messages rate (%), and the Communication Pair priority. Information from the Common Parameters and Communication Settings panes are stored into a configuration file for easy reference.

Upon completion of the benchmarking process, the ERMIS GUI provides user-friendly options for viewing and exporting the results of the experiments. The main output panes are:

- ***Results Table Pane.*** Each row in the result table refers to a specific time interval value and provides data on the number of messages exchanged (within the pre-specified time limit), as well as the number of successful and unsuccessful message cycles. This



Communication Pair
Sender Agent: Agent1
Receiver Agent: Agent2
Unsuccessful Messages(%): 8
Priority: High
Number of Messages: 100
Set Time Intervals
Initial Value: 10
Last Value: 25
Communication Pair: Agent1-Agent2

```
Born agent testMySniffer1-on-Main-Cont
Born agent Agent1 @natassa:1099/JADI
--> Sniffing Agent1 @natassa:1099/JADE
Born agent Agent1 @natassa:1099/JADI
--> Sniffing Agent1 @natassa:1099/JADE
Procedure finished successfully
```

0% message exchange

Time Interval	Successful messages	Unsuccessful messages	% Successful	% Unsuccessful	Communication
10	39	61	39.0	61.0	unsuccessful
16	98	2	98.0	2.0	successful
25	99	1	99.0	1.0	successful

Figure 1: The ERMIS functionality

pane is closely associated to the ***Results Graph Pane.***

- **Synchronization Graph.** It provides an aggregate representation of the synchronization diagrams, for all the time intervals considered. This way the user can easily identify the minimum time threshold that synchronization occurs.

A snapshot of ERMIS "in action", with all its internal windows expanded, is provided in Figure 1. The ERMIS user is able to set up new experiments, or select to load already defined ones (stored *.conf* files). Throughout the whole process he/she is able to monitor communication and timing, produce graphs and start/stop specific communication cycles to focus on specific communication pairs. All data produced are stored in respective *.log* files for later processing.

4. Experiments and Results

A substantial number of experiments was performed in order to validate ERMIS performance, as well as to extract interesting results on the behavior of MAS in soft real-time environments. Scenarios deployed varied on the number of communication pairs, the number of time intervals and the number of messages exchanged for each communication pair. Another parameter that was investigated was the existence or not of common members in different communication pairs, participating either as senders or receivers. Table 1 depicts the performance of small-scale experiments (due to space limitations). Three different configurations are presented: a) a group of agents with no common member in the communications pairs, b) a group of agents with a Common *Sender* and, c) a group of agents with a Common *Receiver*.

All experiments were performed on a typical PC configuration (Intel Pentium4, CPU 2,67 GHz, 512 MS of RAM running Windows XP Home Edition, SP2). As already mentioned, the synchronization point may vary in different configurations, and the ERMIS framework can help at easily identifying the optimal synchronization point. Various time intervals were considered, for testing communication. After enough experimentation, we selected three time intervals T to run all our experiments on: 10*ms*, 16*ms* and 25*ms*. The results of the experiments are discussed next.

Table 1: Successfully monitored messages

Pair	**T = 10ms**	**T = 16ms**	**T = 25ms**
Exp. 1: No Common Members			
Agent1-Agent2	59,4%	90,1%	93,2%
Agent3-Agent4	61,9%	88,1%	92,3%
Exp. 2: Common Sender			
Agent1-Agent2	56,5%	88,3%	92,3%
Agent1-Agent3	74,7%	89,1%	89,8%
Exp. 3: Common Receiver			
Agent2-Agent1	57,3%	85,1%	90,3%
Agent3-Agent1	64,6%	82,0%	90,5%

It is obvious that for T = 10*ms* the performance of the monitoring mechanism proved poor, but it increased drastically for longer time intervals. May one take a close look at the synchronization graphs (omitted due to space limitations), it can be easily identified that groups of agents that have common members in their communication pairs perform worse that in the disjoint couple case. Synchronization between *MySniffer* and the agents is performed for $T \geq 25ms$.

In fact, synchronization is better illustrated in the following experiment. Table 2 summarizes the percentage of the successful message cycles for the three communication pairs (*Agent1–Agent2*, *Agent3–Agent4 and Agent5–Agent6*), denoting that for $T \geq 40ms$ the MAS will be perfecty synchronized with *MySniffer*. This way, developers can easily identify their systems' synchronization threshold.

Table 6: Successfully monitored messages

Pair	T = 10ms	T = 40ms	T = 70ms
Agent1-Agent2	63,2%	99,3%	100%
Agent3-Agent4	68,7%	99,1%	100%
Agent5-Agent6	64,7%	97,0%	100%

5. Conclusions – Future Work

Within the context of this work ERMIS is presented, a framework for monitoring MAS timeliness and for determining the minimum time threshold for a given MAS to perform on time. Given a specific operating environment, ERMIS ensures that the MAS will perform all actions on time, given that all deadlines are above the threshold identified. ERMIS provides the developer with the ability to fully parameterize the problem, and configure his/her own application in order to extract useful conclusions on the optimal operation point of their soft real-time application.

Based on the experimental results, interesting conclusions are drawn: a) it was identified that the synchronization point of MAS is related to the time interval between message cycles, b) it was revealed that agents with multiple roles (i.e. being involved in multiple communication pairs) perform worse than agents with simple roles and, c) for each soft real-time MAS a different synchronization point exists, which is related to a number of factors, i.e. the number of agents in a MAS, the number of communication pairs, the overall communication load of each agents etc.

Future work includes investigation on the effect of *priority*, i.e. the definition of a communication pair as being more important than others (in order to incorporate the notion of important interactions). This way, communication pairs with high priority will be synchronized first, assuring faster response if deemed necessary. Additional future work could be the automation of the process of incorporating the ERMIS mechanism into already deployed MAS. Finally, the synchronization process could be automated, by letting ERMIS define the optimal synchronization point.

6. References

[1] M. Wooldridge, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge , MA ,USA , 1999.

[2] H. Kopetz, and G. Bauer, "The Time-Triggered Architecture", *Proceedings of the IEEE*, Special Issue on Modeling and Design of Embedded Software, vol. 91, no. 1, Jan. 2003, pp. 112-126.

[3] D. J. Musliner, E. H. Durfee, and K. G. Shin, "CIRCA: A Cooperative Intelligent Real-Time Control Architecture", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 6, , Nov/Dec 1993, pp. 1561-1574.

[4] J. Soler, V. Julian, M. Rebollo,C. Carrascosa, and V. Botti ,"Towards a Real-Time Multi-Agent System Architecture", *First International Workshop on Challenges in Open Agent Systems*, Bologna, Italy, July 2002.

[5] I. Soto, M. Garijo, C. A. Iglesias, and M. Ramos, "An Agent Architecture to fulfill Real-Time Requirements", *Proceedings of the fourth international conference on Autonomous agents*, ACM Press NY, Barcelona, Spain, June 2000, pp. 475-482.

[6] D. D. Zeng, and K. P. Sycara , "Agent-Facilitated Real-Time Flexible Supply Chain Structuring", *Proceedings of Agents 99 Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain*, Seattle,Washington, May 1999, pp. 21-28.

[7] L. Cingiser DiPippo , V. Fay-Wolfe, L. Nair ,E. Hodys, and O. Uvarov, "A Real-Time Multi-Agent System Architecture for E-Commerce Applications", *ISADS Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems*, IEEE Computer Society, March 2001, pp. 357-364.

[8] L. S. Nair, "Extending ACL to support communication in a real-time multi-agent system", *Technical Report*, TR00-279, Rhode Island, Dec. 2000.

[9] K. Jurasovic, G. Jezic, and M. Kusek , "A Performance Analysis of Multi-Agent Systems", *International Transactions on Systems Science and Applications*, Special Issue Section on Multi-Agent System Technologies ,vol. 1, no. 4, September 2006, pp. 335-342.

[10] K. Chmiel, D. Tomiak, M. Gawinecki, P. Karczmarek, M. Szymczak, and M. Paprzycki, "Testing the Efficiency of JADE Agent Platform", *Proceedings of the ISPDC/HeteroPar'04*, Cork, Ireland, July 2004, pp. 49-56.

[11] G. Vitaglione, F. Quarta, and E. Cortese, "Scalability and Performance of JADE Message Transport System", Proceeding of the *AAMAS Conference*, Bologna, 2002.