Journal of Universal Computer Science, vol. 14, no. 7 (2008), 1154-1178
submitted: 1/10/07, accepted: 21/1/08, appeared: 1/4/08 © J.UCS

Structural Performance Evaluation of Multi-Agent Systems

Dariusz Król
(Institute of Applied Informatics
Wrocław University of Technology, Poland
Dariusz.Krol@pwr.wroc.pl)

Michał Zelmozer
(Faculty of Computer Science and Management
Wrocław University of Technology, Poland
(m_zelmer@o2.pl)

Abstract: This paper is dedicated to the issue of structural performance of multi-agent platforms. Due to the wide range of all available architectures, we have concentrated only on Java RMI implementations. The main goal of this paper consists of two parts. The first one is to investigate and develop the performance metrics to enable evaluation of distributed systems without reorganization of the running system. The second part is the programming verification of two considered Java RMI multi-agent solutions: Aglets and Jade. We have examined the defined metrics in many experiments with different network and environment configurations to provide experimental evidence that these metrics are adequate in variety of conditions.

Keywords: metric, multi-agent system, performance evaluation
Categories: C.2.4, C.4, H.1.0

1 Introduction

This paper touches the issue of distributed system technologies. The significance of distributed systems has grown substantially during the last couple of years. The main advantage of these systems is not only the increased computational power which can be delivered by distributed technologies but also the range of possibilities and improvement that they provide. The main advantages of these systems are scalability, transparency, simultaneousness, robustness and inter-platform operability. They have become more often implemented in commercial and educational solutions. These characteristics are the main reason for insight and detailed investigations into distributed systems. There are of course some drawbacks to these solutions like sensitivity to network bandwidth and vulnerability for numerous kinds of attacks.

Distributed systems are so vast that it is worth organizing them in categories. The first division can be done on a technological basis. The most adequate and competitive examples in this group are P2P, Grid Computing and Distributed Object Systems (DOS). The main topic of this paper is Distributed Object Systems. To make analysis more valuable it is necessary to narrow down this group again and make another division based on architecture. The most competitive and popular examples this time are CORBA, DCOM and Java RMI. The next and last division in this analysis is done by a matter of implementation and the selection from the last group fell on Java RMI developed by Sun Microsystems, respectively. The choice from the numerous implementations developed by many universities, groups and government projects, which are accessible from the Internet, was done to choose Aglets and Jade. The main reason why these two solutions were chosen is their capabilities, quality of documentation and range of provided examples.

The main problem considered in this work is to verify a correlation between measurements done during tests and the performance estimations based exclusively on metrics. The importance of metrics, described in the Section 2, is significant because if they were confirmed by experiments it would allow them to be used to help to valuate many other realizations without actual tests implementations. All of the matters of distributed systems metrics are amplified in this Section. The basic description, the number of equations, graphs and results present the overall basis for further calculations that will be used in comparison with the real test results.

The next part of the paper contains description of implemented experiments used as a basis for verification of the proposed metrics. The general idea of the experiments is to send a message through all of the agents taking part in the experiment but only once through each agent. It is similar to Hamilton's Cycle with the exception that each time an agent receives a message it has to ask agent broker where to send the next message. This basic scenario is complicated by network configurations such as topological distances, combinations of the number of agents from 128 to 1024 and the number of messages simultaneously sent from 1 to 10.

By the variety of these experiments we try to simulate many different environments in which complete system can be placed. The other reason is the need to check how various connection metrics are suited to experiments in different configurations. All of these features are supported with graphs and tables. In Section 3 we can find results of the experiment, calculations of the metrics and the most important part – verification of the proposed metrics.

As the measurements and calculations have proved proposed metrics can be a powerful and helpful tool in the development of distributed systems. They can help to predict how several Java RMI implementations are suited to various network configurations and environments. We present conclusions about Aglets' and Jade's characteristics, placed in Section 4, as well as the impact of network configuration on distributed system performance.

2 Metrics Description in Distributed Systems

Measurement of effectiveness of Distributed Object Systems is a difficult issue because of variability of the distributed system's environment, architecture, used implementations and many other system characteristics [Krol, 08]. It is obvious that in a real project nobody can afford to create one system or application from a couple of different implementations or even architectures and then select the best solution. Another issue concerned in this paper is how the system architecture effects on the time of the tasks' realization.

We propose connection metrics as the cheapest analytical way to settle the best implementation. By the number of various experiments we try to prove the correlation between measurements done during tests and calculations based on these metrics.

Metrics, described below, were confirmed by experiments. It allows us to treat them as a base for future valuations of different projects without implementing the system.

The main metric that we focus on is a connection cost metric. It allows us to estimate the average distance between agents taking part in our experiment. Metrics assure the high quality of measurement of the distributed systems' effectiveness [Liburne, 04] [Radoslavov, 01]. The key issue is to make metrics measurements in the same conditions that experiments are performed [Gray, 02].

2.1 Connection Metrics

At the outset it is important to understand what is meant by a metric. A metric is a quantitative measure of the degree to which a system possesses a given attribute [IEEE, 90]. A metric is a comparison of two or more measures, for example, body temperature over time. It allows a trend or pattern to be seen in the measure.

For the purpose of calculating time needed to perform the whole assigned work we need to measure the connection cost metrics in a couple of different network configurations - that is time that takes sending a message from an agent on one host to another on different host with a reply. We use formal representation of metrics that ideas were proposed in [Ciobanu, 06]. These metrics take into account quantitative and qualitative elements. Quantitative characteristics refer to elements such as networks, CPUs, or storage. The network element includes four parameters: delay, jitter, packet loss rate and throughput. Qualitative characteristics refer to elements such as host reliability, fault-tolerance and user satisfaction regarding service.

In this paper we consider a network, and a set H of connected hosts. The connection cost metric can be modelled in various ways. For simplicity, we use the functions $L_t : H \rightarrow [0,\infty)$, $istab_t, i\sec_t : H \rightarrow [0,1]$ that represent the average latency, degree of stability, and degree of security, respectively. We want to minimize the host latency, which is defined as the time taken by the host to process received messages. A small latency is crucial to network performance in many aspects and is required by a variety of applications, such as real-time communications based on ad hoc networks. The degree of stability is defined to be the probability that the host can compute tasks successfully. The degree of security is defined to be the probability that the host is secure. All these functions depend on a certain moment of time t. Time is divided into discrete time intervals. All hosts have access to a globally synchronized clock. We can aggregate these aspects into a function $E_t : H \, x \, H \rightarrow [0,\infty)$ defined by

$$E_t(h) = \sqrt{L_t(h)^2 + istab_t(h)^2 + i\sec_t(h)^2} \text{ for all } h \in H .$$

As a distance over H, we use a function $d_t : H \times H \rightarrow [0,\infty)$ defined by

$$\begin{aligned} d_t(h_1,h_2) = [(L_t(h_1) - L_t(h_2))^2 + (istab_t(h_1) - istab_t(h_2))^2 & \\ + (i\sec_t(h_1) - i\sec_t(h_2))^2]^{\frac{1}{2}} & \quad \text{for all } h_1,h_2 \in H . \end{aligned}$$

This function d_t is a pseudo-metric over H and represents the cost of connecting two hosts. The distance function presents only the general idea of measurement of distance between hosts. To make them useful for our experiments we had to add some features that will make them more tangible. In the next section we describe our own approach to the issue of host distance measurement in a working network. Afterwards we present the equations that we have developed to calculate proposed metric and fit them better in different experiment scenarios. The number of variables and parameters that are taken into account were selected in the pre experimental tests.

2.2 Network Configurations

As a connection metric, we propose the measurement of time taken to exchange a message between two agents. We skip over the degree of stability and degree of security. All of the metrics measure only the basic behaviours of the agents because agent does not perform any other action than handling the message and sending the response. Time is measured from the moment when a message is sent until the reply is received. After the reply comes the time of this exchange is divided by 2. That is why results are dependent on both machines: the sender and the receiver as well as on the network usage during the time of measurement. We distinguish three types of configurations of message sent between two agents on: single host, two local network hosts, and two remote hosts.

- **Single host**

In this case the message is sent between two agents on the same host. Results of a metric mainly depend on the machine hardware configuration, current memory and processor usage. In Distributed Object System it is a rather rarely seen situation, but in our experiments it is used as a baseline comparison for more complicated configurations. This case is presented in Figure 1.

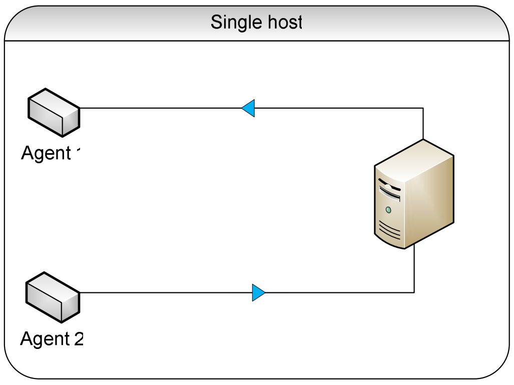

Figure 1: Single host topology of agent cooperation

- **Two local network hosts**

In this case the message is sent between two agents on two different hosts in a local network. Time of the connection between these agents depend mainly on machine hardware configurations, local network speed, usage by other users and configuration of both machines. This case is presented in Figure 2.

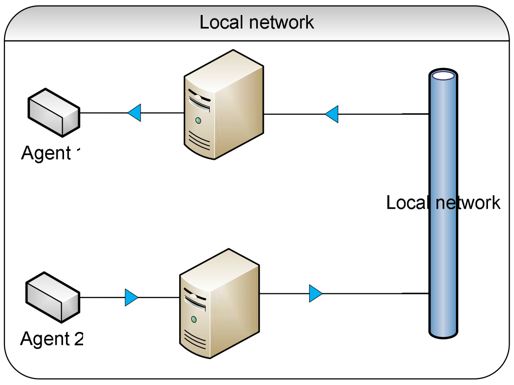

Figure 2: Local network topology of agent cooperation

- **Two remote hosts**

This metric is used to measure time of connection between remote hosts. We send message between two agents on two different hosts in remote locations separated by Internet. Despite great distance of the hosts, Internet connection speed on both ends does not have major importance because one message exchange does not come with transport of a large amount of data. This case is presented in Figure 3.

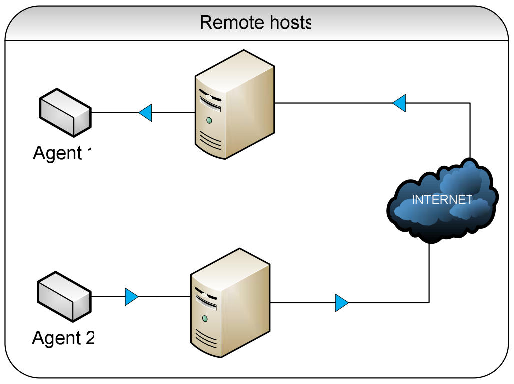

Figure 3: Remote hosts topology of agent cooperation

- **Two remote hosts with firewall**

The message is sent between two agents on two different hosts in remote networks in the presence of a firewall. This instance is similar to the previous one, with the exception of a firewall being placed on one of the host. We can measure how a higher level of security can inflict the time of the connection. In some cases this metric could be not too accurate because of the way firewalls works. The first connection usually takes much more time than the following. That is why all of the measurements will be done repeatedly. This case is presented in Figure 4.

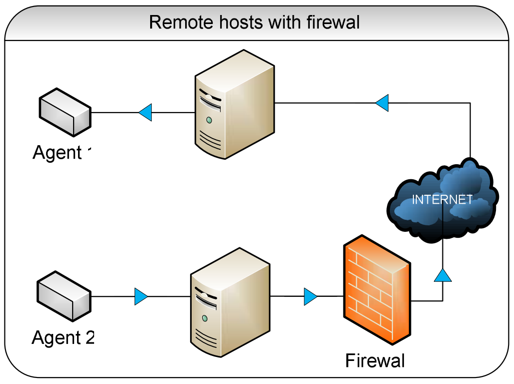

Figure 4: Remote hosts with firewall topology of agent cooperation

3 Experiments

In this Section we present all of the presumptions of our system. We have been based on the traditional definition of multi-agent systems. All of the agents are self-organized but they are coordinated by one agent. The dependency on the agent manager lies only in a work delegation. All of the services available in the system are evenly shared between all of the requests received from agents. Agents are able to perform many other actions without knowledge of the agent broker.

The main goal of the experiments presented here is to valuate metrics described in the previous Section. We try many different tests scenarios to find out how characteristics of the different architectures influence on the performance results and how it copes with calculations of the connection metrics. The other reason for making test is comparison of two different multi-agent implementations: Aglets and Jade. It would be crucial to find the differences between these two architectures mainly in effectiveness and scalability. Below we propose some experiment scenarios and network configurations.

The first part of this Section is devoted to the presentation of performed experiment scenarios and measured results. Then we present the calculated results of the metrics described in the previous Section. The last part is the comparison of these two parts: tests results and metrics calculations.

3.1 Communication Process

Unified process of communication is presented in Figure 5. There are eight steps in the single communication act between three agents:

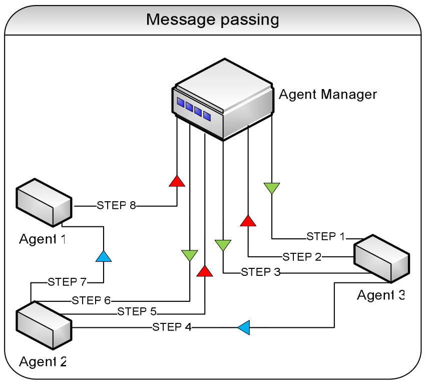

Figure 5: Process of communication between agents

1 Agent Manager randomly selects one agent from all of the agents (e.g. Agent 3) and sends it the "get next" message.
2 When Agent 3 receives "get next" message it sends "get next" message to the Agent Manager to get the IP address of the host and name of next agent.
3 Agent Manager sends the reply to Agent 3. Agent 3 receives the reply from the Agent Manager with the coordinates of the next agent.
4 Agent 3 sends message "move on" to the next agent – e.g. Agent 2.
5 Agent 2 receives message from Agent 3 and sends message "get next" to the Agent Manager.
6 Agent Manager sends the message to Agent 2. Agent 2 receives the reply from the Agent Manager with the coordinates of the next agent.
7 Agent 2 sends message "move on" to the next agent - Agent 1.
8 Agent 1 receives message from Agent 2 and sends message "get next" to the Agent Manager. Agent Manager ends the process.

3.2 Experiments Characteristics

There are four main characteristics of the experiments that are combined with each other to reach the number of different configurations:

1. Different network configurations of the host: local network, remote locations, the presence of a firewall. We try to find out how different network configurations affect the experiments results and how Aglets and Jade cope with different configurations.
2. The number of hosts taking part in an experiment varies from one to three: one *home* computer with ADSL connection and two computers: *zsi.103*, *zsi.105* placed in the local network at the university area. We try to observe how distribution of agents between hosts can make a system more efficient (e.g. what is better two hosts with 100 agents on each host or one host with 200 agents).
3. The total number of agents taking part in an experiment is selected from the set: 128, 512, and 1024. We try to compare work of the same amount of

agents distributed on a number of different network configurations. We also observe how a rising number of agents affect machine usage.

4. The number of messages sent is selected from the set: 1, 2, 5, and 10. We try to incriminate the hosts not only by increasing the number of agents but also by sending simultaneous messages. We want to find out how concurrent agents share processor, memory and network resources of the host computer.

3.3 Experiments Configuration

Experiments are organized in categories. All of the categories that are presented below come as a result from combination of the total number of agents and the number of sent messages with or without firewall. The metrics were evaluated having as factors the number of agents (dented as *NA*) and the number of messages (denoted as *NM*). The expressions such as *zsi.103_home*, *zsi.105_zsi.103* means the connection time between two hosts.

3.3.1 Connection without Firewall

- **Single host**

This is the simplest configuration: Agent Manager and all of the agents are placed on the same host. There is no communication through the network. This configuration is presented in Figure 6.

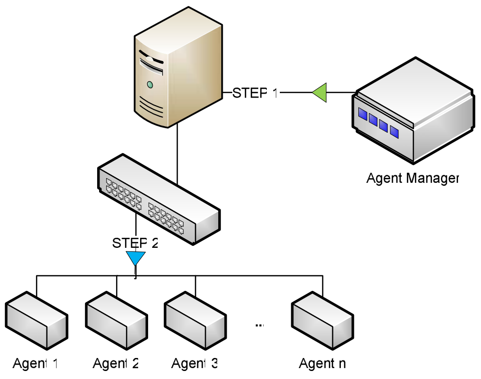

Figure 6: Single host configuration

Appropriate connection metrics for this configuration and for the communication schema between agents (see Figure 5) are shown in Table 1.

home	*NA*NM*[3*home_home]*
zsi.103	*NA*NM*[3*zsi.103_zsi.103]*
zsi.105	*NA*NM*[3*zsi.105_zsi.103]*

Table 1: Single host – broker on the same host

- **Two hosts on the local network**

Just like on the previous configuration Agent Manager shares the host with the half of the agents, but there is another host placed in local network with other half of the agents. This scenario is presented in Figure 7.

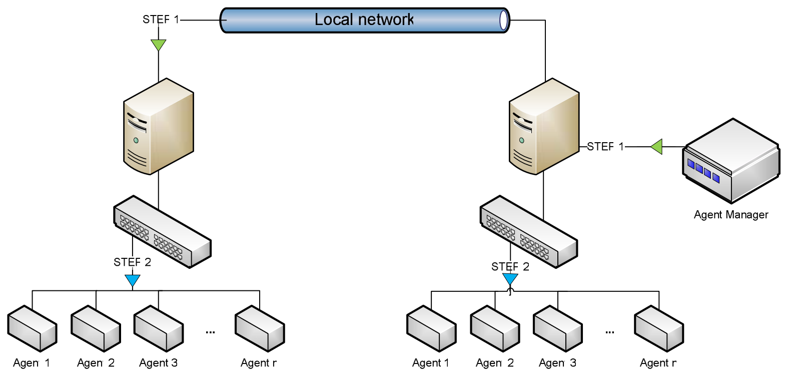

Figure 7: Configuration of two local hosts

There are two kinds of scenarios: worst and best. The worst scenario (dented as WS) is when agents communicate one by one from two hosts through the network, so there is much longer time of information exchange. The best scenario (dented as BS) is an ideal situation when all of the agents on first host communicate with each other at first and then all of the agents on the second host. Broker can be placed on two hosts either on *zsi.103* or *zsi.105*. In this case the best scenario and the worst scenario metrics are shown in Table 2 and Table 3. The number of messages is divided by 2 because there are two hosts.

WS	*NA*NM*[2*zsi.103_zsi.105+zsi.105_zsi.105]*
BS	*NA*NM*[zsi.103_zsi.105+3/2*zsi.105_zsi.105+1/2*zsi.103_zsi.103]*

Table 2: Two local hosts – broker on zsi.105

WS	*NA*NM*[2*zsi.103_zsi.105+zsi.103_zsi.103]*
BS	*NA*NM*[zsi.103_zsi.105+3/2*zsi.103_zsi.103+1/2*zsi.105_zsi.105]*

Table 3: Two local hosts - broker on zsi.103

- **Two hosts on remote locations with local broker**

This configuration is similar to the previous one: Agent Manager shares the host with other agents; half of the agents are placed in the remote location. This configuration is presented in Figure 8. There are two kinds of scenarios: worst and best. The worst scenario is when agents communicate one by one between two hosts, so there is much longer time needed. The best scenario is an ideal situation when the agent on one host communicates with each other and then all of the agents on the other host send the messages. Broker was placed on remote host and the second host taking part in communication can be either *zsi.103* or *zsi.105*. In this case the best scenario and the worst scenario metrics are presented in Table 4 and Table 5.

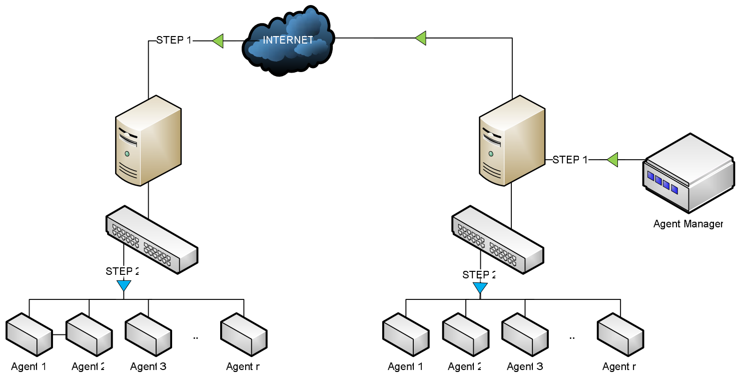

Figure 8: Configuration of two remote hosts

WS	*NA*NM*[2*zsi.103_home+home_home]*
BS	*NA*NM*[zsi.103_home+3/2*home_home+1/2*zsi.103_zsi.103]*

Table 4: Two remote hosts - broker on home and zsi.103

WS	*NA*NM*[2*zsi.105_home+home_home]*
BS	*NA*NM*[zsi.105_home+3/2*home_home+1/2*zsi.105_zsi.105]*

Table 5: Two remote hosts – broker on home and zsi.105

- **Two local hosts with remote broker**

This configuration is similar to the previous one, but this time Agent Manager does not share the host with other agents. There are two hosts in local network with half of agents on each of them and Agent Manager is placed on the remote host. This scenario is presented in Figure 9. As it was before there are two kinds of scenarios: worst and best. Agents in best variant can exchange messages on one host and then on the other host. The best scenario and the worst scenario metrics are shown in Table 6.

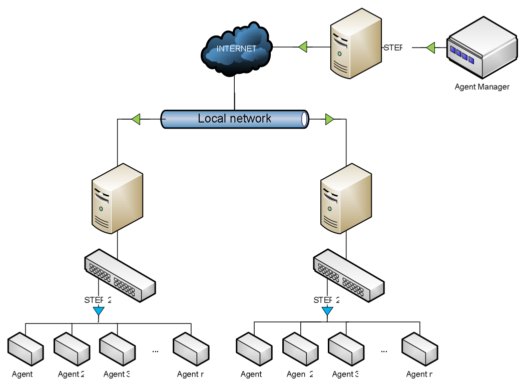

Figure 9: Configuration of two local hosts with remote broker

WS	*NA∗NM∗[zsi.103_home+ zsi.105_home+zsi.103_zsi.105]*
BS	*NA∗NM∗[zsi.103_home+ zsi.105_home+1/2∗zsi.103_zsi.103+1/2∗zsi.105_zsi.105]*

Table 6: Two local hosts – broker on home

- **Host with local broker and remote host**

This configuration is similar to the previous one. Agent Manager is placed in the local network on one host, on the second local host the first half of the agents are running; the other half is placed on the remote host. This configuration is presented in Figure 10. There are two kinds of scenarios: worst and best and nothing have changed in that matter. Broker is placed on *zsi.103*, the other hosts taking part are remote *home* host and zsi.105. In this case the best scenario and the worst scenario metrics are shown in Table 7.

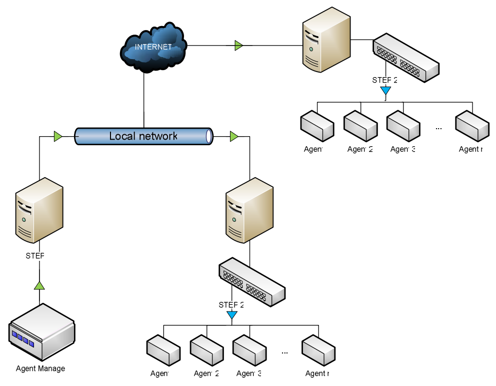

Figure 10: Configuration of host with local broker and remote host

WS	*NA*NM*[zsi.103_home+ zsi.105_home+zsi.103_zsi.105]*
BS	*NA*NM*[zsi.103_home+ zsi.103_zsi.105+1/2*home_home+1/2*zsi.105_zsi.105]*

Table 7: Host with local broker and remote host – broker on zsi.103

3.3.2 Connection with Firewall

In order to check how high level of security can inflict the communication metrics and overall results of the experiments there were prepared three configurations with presence of firewall.

- **Two hosts on remote locations with local broker**

This configuration is similar to the configuration of respective scenario without firewall. It is presented in Figure 11. Equations are the same as it was in configuration without firewall, but the values of connection time are changed. One host is placed in remote locations and again the other host taking part in communication can be either *zsi.103* or *zsi.105*. In this case the best scenario and the worst scenario metrics are shown in Table 4 and Table 5.

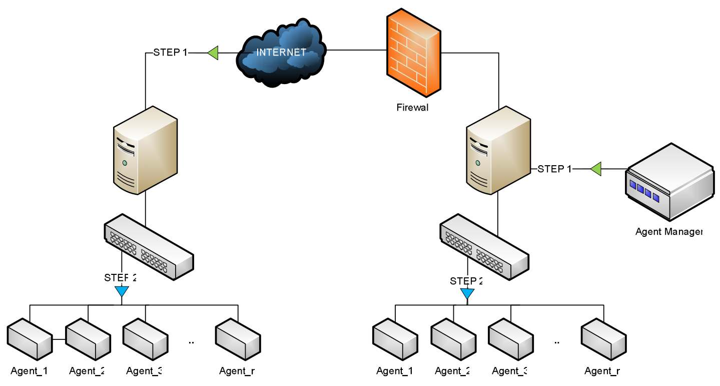

Figure 11: Configuration of two remote hosts with local broker

- **Two local hosts with remote broker**

This configuration is similar to the one without firewall and the equations are the same as it was in configuration without firewall. It is presented in Figure 12. One host is placed in remote locations and again the other hosts taking part in communication are zsi.103 or zsi.105. The best scenario and the worst scenario metrics are in Table 6.

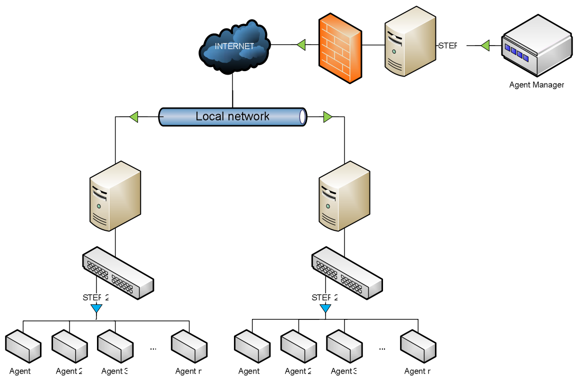

Figure 12: Configuration of two local hosts with remote broker

- **Host with local broker and remote host**

This configuration is similar to the configuration without firewall. This configuration is presented in Figure 13. As it was before there are two kinds of scenarios: worst and best. Broker is placed on *zsi.103*, the other hosts taking part are home host and *zsi.105*. In this case the best scenario and the worst scenario metrics are in Table 7.

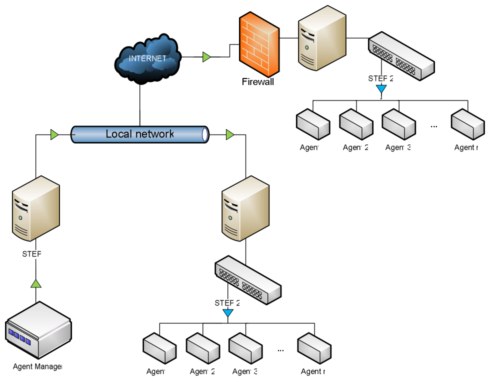

Figure 13: Configuration of one host with local broker and remote host

3.4 Verification of Proposed Metrics

In this Section we focus on the main issue of this paper. We will compare WS and BS metrics in two extra conditions: low duty (dented as LD) and heavy duty (dented as HD) for Aglets and Jade with measured results. First one denotes the connection time when system usage is very low, second one is measured in highest load of the system. In this case additional 1024 agents are running on hosts.

3.4.1 Connection without Firewall

- **Single host**

To compare metrics with results we select combination of small and large number of agents (128, 1024) and messages (2, 10). There is only one possible scenario in this experiment, because only one host takes part in it.

First we will compare configuration with small number of agents for both architectures. In Figure 14 and Figure 15 we can see two broken lines and one continuous presenting suitably metrics and result. As we can see measured results of both architectures Aglets and Jade are close to low duty metric. Results are much lower when the number of simultaneous messages is higher.

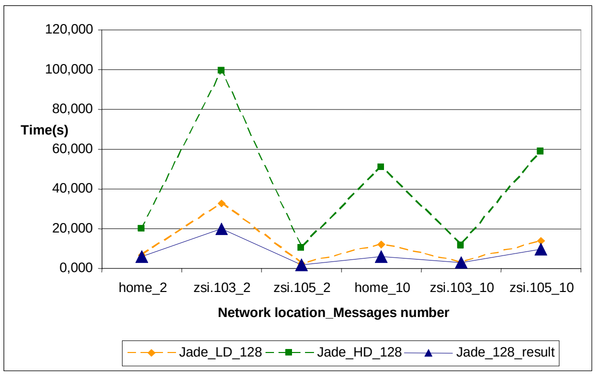

Figure 14: Jade - 128 agents

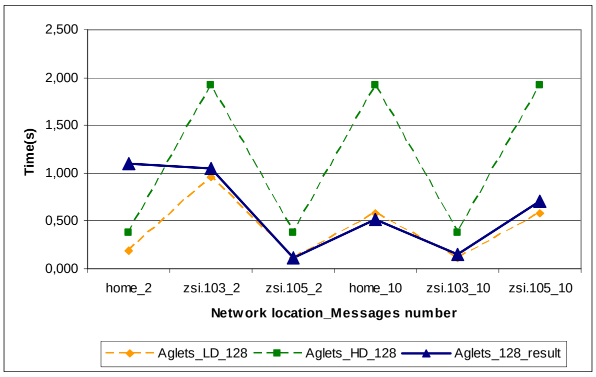

Figure 15: Aglets - 128 agents

In Figure 16 and Figure 17 we can see how metrics work if we consider higher number of agents. As it was for 128 agents, results of Aglets and Jade are close to low duty metric.

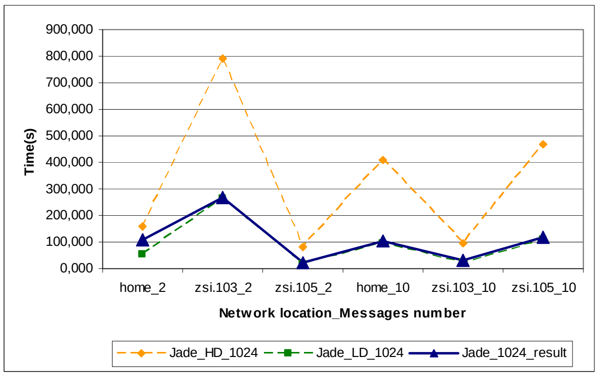

Figure 16: Jade - 1024 agents

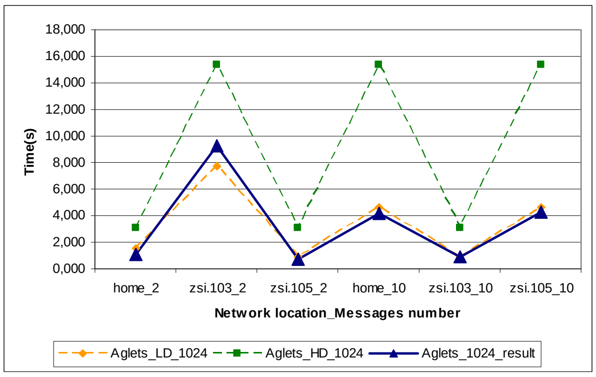

Figure 17: Aglets - 1024 agents

- **Two local hosts**

Below in Figure 18 and Figure 19 both have logarithmic representation of time. There are presented metrics for two local network hosts. In this scenario there are four different metrics as a combination of low and heavy duty environment and the best and the worst scenario. As we can see Aglets' as well as Jade's measured results are very close to calculated metrics.

Jade results for configuration of ten messages are below the lowest metric, but for one and two messages it fits well between drawn metrics. Generally Jade stays close to lower metrics which are low duty metrics. Type of scenario in this case is not so crucial because hosts are very close to each other.

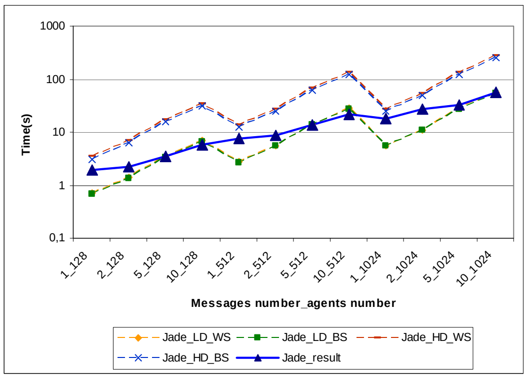

Figure 18: Jade - two local hosts

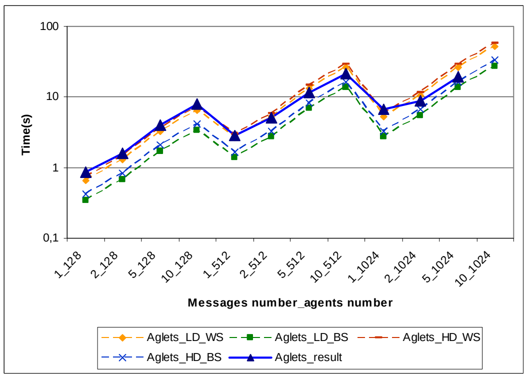

Figure 19: Aglets - two local hosts

Aglets' results for 128 are close to the highest metric but with the rise of agents number results are almost in the middle between low and heavy duty metrics. Again type of the scenario does not have big influence on results because hosts are in local network. It is important to point out that Aglets failed to exchange ten messages between 1024 agents in couple of tests.

- **Two remote hosts**

Results and metrics for two remote hosts are presented in Figure 20 and Figure 21. In this case when hosts are in distant locations type of scenario has even bigger influence on metrics than level of system usage.

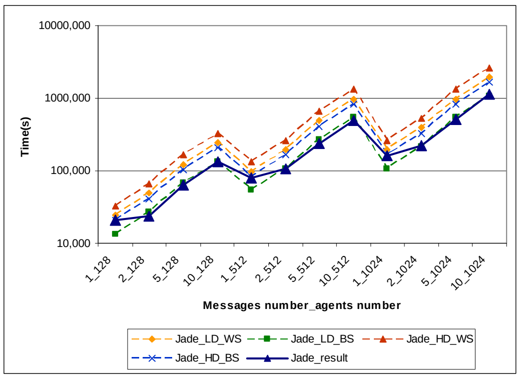

Figure 20: Jade - two remote hosts

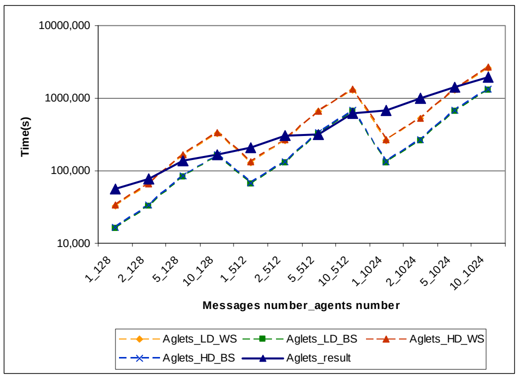

Figure 21: Aglets - two remote hosts

- **Two local hosts with remote broker**

Results and metrics for two local hosts with remote broker are presented in Figure 22 and Figure 23. In this instance there are four different combinations of low and heavy duty environment and the best and the worst scenario. As we can see Jade's results are very close to calculated metrics, almost all results fit between lowest and highest metrics. In this case type of scenario seems not to be crucial.

The configuration with Aglets presents a bit worse dependence. In this case and only in this case Aglets results were much higher than any of calculated metrics. The only explanation can be found in metrics calculations, because they are all almost equal. This is the only case when any of metrics fail to cope with measured results so widely.

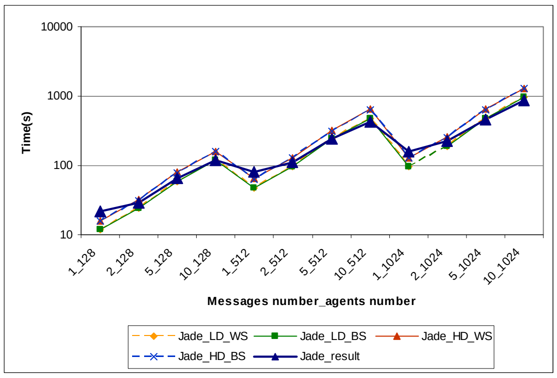

Figure 22: Jade - two local network hosts with remote broker

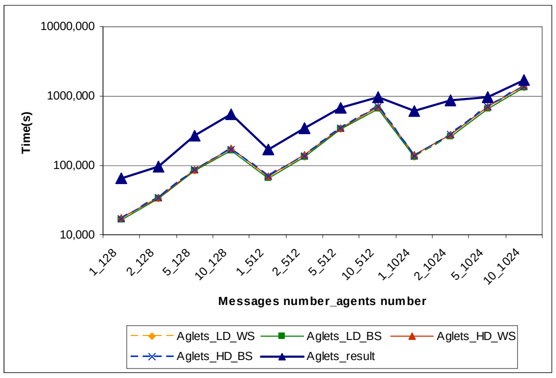

Figure 23: Aglets - two local network hosts with remote broker

- **Host with local broker and remote host**

Results and metrics for host with local broker and remote host are presented in Figure 24 and Figure 25. In this configuration there are four different metrics as a combination of low and heavy duty environment and the best and the worst scenario. As we can see Jade's results are very close to calculated metrics, almost all results fit to lowest highest metrics. In this case quality of scenario (best, worse) seems to be as important as environment usage (low, heavy) because they stay in similar distance for all cases.

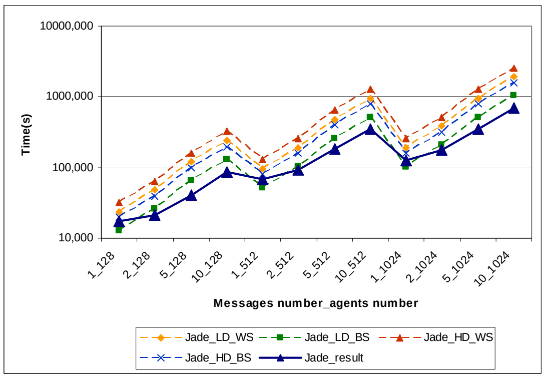

Figure 24: Jade - one host with local broker and remote host

If we consider Jade's results as it was with two remote hosts scenario it is easy to notice that they stay close to low duty and the best scenario metric. In some configurations results are close to heavy duty and the best scenario metric.

Again Aglets' metrics seem not to be so affectionate by low or heavy duty scenario. Here the tendency is a bit different, Aglets result rather stay close to higher metrics and even exceed them in high number of agents configuration.

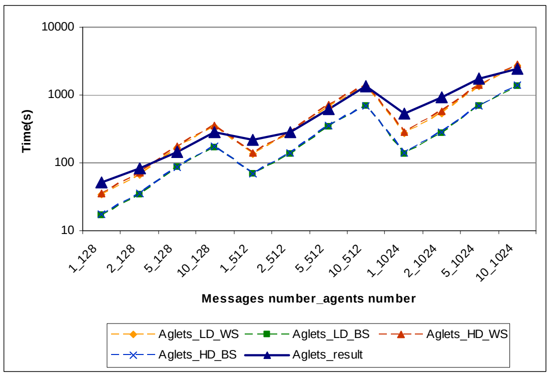

Figure 25: Aglets - one host with local broker and remote host

3.4.2 Connection with Firewall

- **Two remote hosts**

Results and metrics for two remote hosts with firewall are presented in Figure 26 and Figure 27. These results are very close to results of the same configuration without firewall.

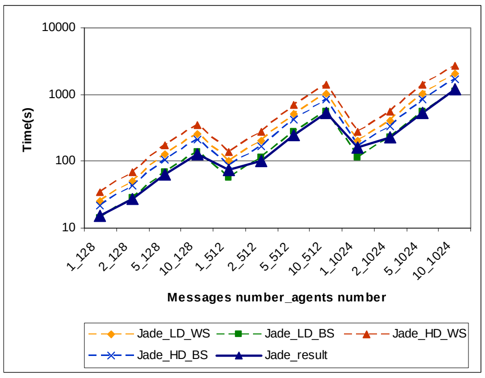

Figure 26: Jade - two remote hosts with firewall

If we consider Jade's results it is easy to notice that they stay close to low duty best scenario metric. In some configurations results are close to heavy duty best scenario metric.

Aglets' metrics do not seem to be influenced by low or heavy duty scenario. Here tendency is a bit different, Aglets results rather stay close to higher metrics and even exceed them in low number of messages configuration as it was without firewall. With rising number of messages and agents Aglets' results seem to fit better to metrics.

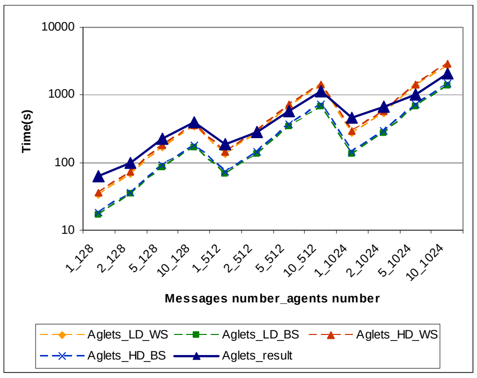

Figure 27: Aglets - two remote hosts with firewall

- **Two local hosts with remote broker**

Results and metrics for two local hosts with remote broker with firewall are presented in Figure 28 and Figure 29.

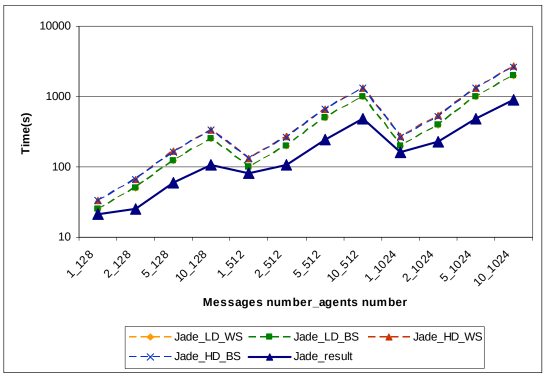

Figure 28: Jade - two local hosts and remote broker with firewall

These results are very close to results of the same configuration without firewall. Graphs of Jade are very similar to these without firewall. It is very interesting that when firewall is involved in experiment, despite the fact that results are higher than without it, metric still fits to results. If Aglets are considered metrics fit better than it was in scenario without firewall. Other interesting thing is that all of the metrics are almost equal, just like it was in scenario without firewall.

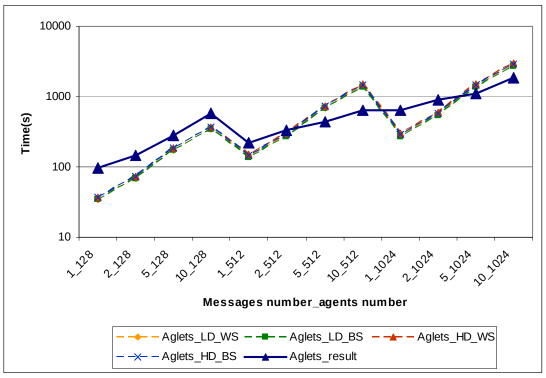

Figure 29: Aglets - two local hosts and remote broker with firewall

- **Host with local broker and remote host**

Results and metrics for host with local broker and remote host with firewall are presented in Figure 30 and Figure 31. These results are very close to results of the same configuration without firewall.

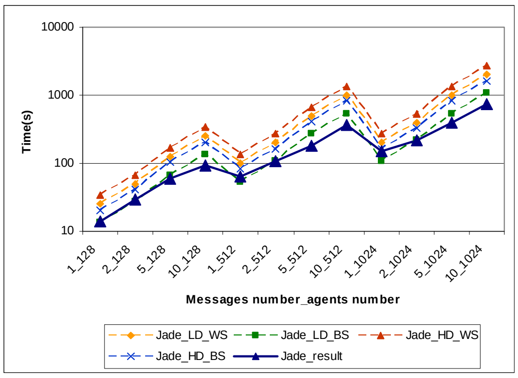

Figure 30: Jade - one host with local broker and remote host

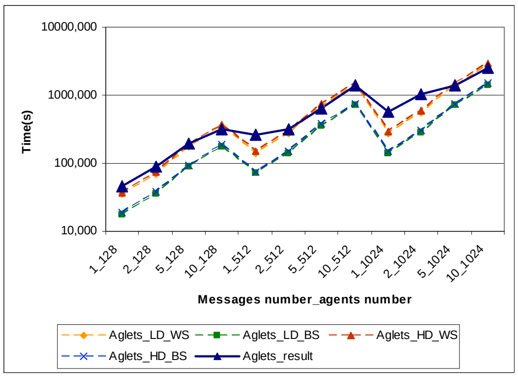

Figure 31: Aglets - one host with local broker and remote host

3.4.3 Verification Summary

Through summarizing metrics verification we try to generalize how these metrics and pre experiment assumptions compare to our experimental results. Low duty and heavy duty results combined with the best and worse scenarios metrics have created a range that can quite accurately predict how the real system implementation will behave. One of the Java RMI-based implementations (Jade) has good performance and generally better fits to the metrics. Despite the fact that Aglets behave a bit worse in predictions, the results of verification are still fair.

To improve calculations we had to take into account the number of messages sent. But it also shows that both of the architectures have some weaknesses in simultaneous messages exchange.

Agent technology promises to build the cost-effective, distributed systems that are powerful and flexible. However, several problems have emerged what prevents multi-agent systems from being applicable in many real-world settings. A major flaw of this technology is the use of the appropriate protocols, what makes it difficult for agents that have not been designed to work together to interoperate. Because the Web services technologies arose as the best solution for remote execution of functionality, the Service-Oriented Architecture (SOA) concepts could be applied to integrate and utilize the proposed metrics effectively. SOA is the evolution of other solutions in the distributed programming field such as RMI, CORBA or DCOM [SUN, 08]. The utilization of standard protocols enables web services to constitute loosely coupled distributed systems. Communication between two services relies on an asymmetric interaction based on request/reply message exchange. Our prototype verification system could be developed using e.g. GlassFish – the application server with Java API for Web Services (JAX-WS) 2.0 to send messages over the Internet. We believe that the combination of Web services and agents provides a promising computing paradigm for efficient distributed messages processing. As a proof of concept of the technologies discussed above, there have been designed a system which demonstrates the integration of Jade with Web Services [Bellifemine, 08].

4 Conclusions and Future Work

The main contribution of this paper is based on proving the statement that the proposed metrics allow to anticipate distributed system characteristics and predict the time of the job completion. It is obvious that we cannot predict the exact time that a system needs for the planned work. By the number of metrics that comply with the lowest and highest environment load as well as the worst and the best scenarios of the job, we can predict close approximations of what results the real system will accomplish. The crucial issue is the selection of adequate single metrics measurements.

All of the developed experiments have proven the main properties of Distributed Object Systems like transparency, scalability and robustness. The easy way to exemplify the level of complexity of the experiments is to point out that the most complicated scenario consisted of 1024 agents placed on three hosts in different locations that have exchanged 10 messages between each other simultaneously. Concurrency in messaging is the only feature of these solutions that has not been fully satisfied in all experiment. The number of simultaneously transmitted messages has an almost linear influence on the time of all messages exchanged. It does not prove that these solutions are not concurrent in other matters but it shows that message exchange has some constraints. It is probably caused by weak messaging protocols that are shared by all agents which generate queues.

If we look at the presented experiment we may make some conclusions about the impact of the network configuration that agents were placed in. If one of the hardest experiment configurations is considered, as it was mentioned before 1024 agents and 10 messages, configuration of two hosts in local network presents the best results. If we consider architecture including remote hosts, configuration of one host with local network broker and remote host has accomplished better results than two remote hosts' configurations and two local hosts with remote broker. This shows that distribution of the work is crucial in this kind of solutions. The other thing that has been proved is the slight influence of firewall on results. We can observe just minimal but regular increase in all compared tests but it did not change the accuracy of the metrics.

In the future this work could be expanded in three directions. One of them is to explore higher level of task complexity performed by agents; this will enable checking if these metrics are adequate for complicated calculations as well. The second direction is the extension of comparison between other architectures or even technologies that could show if this work concerns other solutions. The third direction could be dedicated to find other suitable multi-agent metrics that would help in making correct predictions.

Acknowledgement

The authors would like to thank Gabriel Ciobanu from "A.I.Cuza" University of Iasi for discussions that became the inspiration for this research. Many thanks to Mariusz Nowostawski from University of Otago who made some comments on the paper.

References

[Bellifemine, 08] Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE: A software framework for developing multi-agent applications. Lessons learned, Information and Software Technology, 50, 2008, 10-21.

[Ciobanu, 06] Ciobanu, G.: Coordination and Self-Organization in Multi-Agent Systems, In Proc. of the 6th International Conference on Intelligent Systems Design and Applications (ISDA 2006), Jinan, China, 2006, plenary lecture.

[Frelechoux, 00] Frelechoux, L., Osborne, M., Haas, R.: Topology optimization of IP over ATM, In Proc. of the Universal Multiservice Networks (ECUMN 2000), Colmar, France, 2000, 122-131.

[Godoy, 04] Godoy, F., Rodriguez, A.: Defining and Comparing Content Measures of Topological Relations, Geoinformatica, 8(4), 2004, 347-371.

[Grama, 93] Grama, A., Gupta, A., Kumar, V.: Isoefficiency Function: A Scalability Metric for Parallel Algorithms and Architectures, IEEE parallel and distributed technology: systems and applications, 1(3), 1993 12-21.

[Gray, 02] Gray, R., et al.: D'Agents: Applications and Performance of a Mobile-Agent System, Software: Pracice and Expirience, 32(6), 2002, 543-573.

[Gruer, 02] Gruer, P., Hilaire, V., Koukam, A., Cetnarowicz, K.: A formal framework for multi-agent systems analysis and design, Expert Systems with Applications, 23, 2002, 349-355.

[Haeuser, 00] Haeuser, J., et al., A test suite for high-performance parallel Java, Advances in Engineering Software, 31, 2000, 687-696.

[IEEE, 90] IEEE standard glossary of software engineering terminology. IEEE Std. 610.12, 1990.

[Krol, 08] Król, D., Zelmozer, M.: A Comparison of Performance-Evaluating Strategies for Data Exchange in Multi-agent System, In Proc. KES-AMSTA 2008, LNAI 4953, 2008, 793-802.

[Liburne, 04] Liburne, B., et al., Measuring Quality Metrics for Web Applications, In Innovations through Information Technology, Idea Group Inc. 2004.

[Radoslavov, 01] Radoslavov, P.: The Relationship Between Topology and Protocol Performance: Case Studies, PhD Dissertation, Computer Science Department, University of Southern California, December 2001.

[SUN, 08] Web Services for the Java Platform, available from:
http://java.sun.com/webservices/index.jsp [cited 2008 April 30].

[Shehory, 98] Shehory, O., Kraus, S.: Methods for Task Allocation via Agent Coalition Formation, Artificial Intelligence, 101(1-2), 1998, 165-200.