Emergent Dynamic Routing Using Intelligent Agents in Mobile Computing

Florin Leon, Mihai Horia Zaharia, Dan Gâlea

"Gh. Asachi" Technical University of Iaşi

Faculty of Computer Science and Engineering

e-mail: florinleon@gmail.com

Abstract: Routing begins to have an important place in the context of high performance distributed systems, with an increasingly notable influence on the overall performance of the system under analysis. While many global algorithms have been proposed for the routing problem, in this paper we demonstrate how a relatively simple agent-based approach, based on ideas inspired from complex systems and reinforcement learning, can generate a highly complex set of local interactions between individual agents, whose emergent behaviour results in the desired routing effect.

Keywords: emergence, routing, intelligent agents, mobile computing

Florin Leon holds a PhD in Computer Science and is presently a Lecturer in the Department of Computer Science and Engineering of the "Gh. Asachi" Technical University of Iaşi. His main research interests are: artificial intelligence, simulations using intelligent agents, and data mining. He is the author or co-author of 7 books and over 50 articles.

Mihai Horia Zaharia is an Associate Professor in the Department of Computer Science and Engineering of the "Gh. Asachi" Technical University of Iaşi and has a PhD in distributed systems. His main research areas are: distributed computing, distributed artificial intelligence, knowledge-based geographical systems, computer systems security, and computer architectures. He is the author or co-author of 9 books and over 40 articles.

Dan Gâlea is a Professor in the Department of Computer Science and Engineering of the "Gh. Asachi" Technical University of Iaşi, and a PhD Supervisor. His main research areas are: artificial intelligence, knowledge-based geographical systems, image processing, and natural language processing. He is the author or co-author of 10 books and over 100 articles.

1. Introduction

Nowadays, one of the important problems of information and communications technology is related to re-designing interactions between isolated computing elements in order to create large, powerful, and fault tolerant distributed systems. There are already many approaches that try to solve this problem, from e-government to virtual universities and organizations, peer to peer communities, GRID computing, and e-science computing. If the approaches offered by the traditional engineering cannot keep up with this convergence, those conglomerates of computing power can create many problems due to an inherent instability and mostly to their non-predictive behaviour if some limits are reached. The analysis of the natural complex systems leads us to alternative approaches. The examples under scrutiny refer to animal brain, immunity systems, ants colonies, and so on. All these examples have powerful, complex, aggregated behaviours obtained from the combination and interactions of simple elements or individuals. As a result, this kind of problem solving techniques begins to be used for various specific problems.

Distributed systems are usually analyzed from the points of view of load balancing, geographical distribution of the control, and databases. In most models, the communications are presumed to be very good and fault tolerant. Most of the time, these attributes are valuable but in some cases there are serious quality of service degradation at the communication line hot spots where bottlenecks appear due to the used routing algorithms. Therefore the routing problem begins to have equal importance with the previously mentioned basic characteristics of distributed systems.

In this paper a distributed framework based on intelligent agents was used to propose and test a new approach in routing for mobile devices. According to Wooldridge (2000), an agent is a computer system that is situated in its environment and is capable of autonomous action in order to meet its design objectives. Agent Oriented Programming, AOP, is a fairly new programming paradigm that supports a societal view of computation. In AOP, objects known as agents interact to achieve individual goals. Agents can exist in a structure as complex as a global internet or one as simple as a module of a common program. Agents can be autonomous entities, deciding their next step without the interference of a user, or they can be controllable, serving as an intermediary between the user and another agent.

Intelligent agents retain the properties of autonomous agents, and in addition show a so-called "flexible" behaviour (Wooldridge & Jennings, 1995): reactivity (the ability to perceive their environment, and respond in a timely fashion to changes that occur in it), pro-activeness (the ability to exhibit goal-directed behaviour by taking the initiative), and social ability (to interact with other agents and possibly humans).

2. Routing in Mobile Computing

Due to the inherent mobility of the nodes, the routing protocols at both the ad hoc and internet network levels are different from the classical approach. As a result, proactive schemes are used to continuously improve the routing tables for mobile nodes. Unfortunately, they use large amounts of available broadband and sometimes computing power. On the other hand, the on-demand routing protocols spends a lot of time for route discovery and they are sometime not suitable for real-time communication or processing.

There is a third approach which implies mobile agents. They can be used to establish an efficient route and also topology discovery, and increase the node connectivity (Marwaha, Tham & Srinivasan, 2005). The agent-based approach has some disadvantages related to the total node dependencies on knowledge provided by the agent. It is possible that routing can fail when the network topology is dynamic and the time to live for a route is short. In agent-based routing, the mobile nodes must wait until the routing agent provides the required routes to begin a communication. In some situations, it is possible that the nodes that carry the agents to be unexpectedly disconnected from the network. The causes may vary from a too long radio signal to entering the sleep mode or shut down. With any closed carrying node, the number of routing agents will decrease and so will the routing efficiency

There are numerous ad-hoc routing algorithms that accept various conditions to connect in the network. They can be classified as proactive or reactive. As we previously mentioned, the proactive ones are more efficient but sometimes they can generate unjustified overload of computing and communication resources, and it may be an idea to create route only on demand like the other class of algorithms. The reactive approach can produce undesired delays with route establishment, especially when they cannot reach the destination and usually a proactive approach is preferred (Carrillo, Marzo, Vilà & Mantilla, 2004). Even if the proactive protocols offer a better image of the network when the topology is highly dynamic, the routing effort also creates overloads in communication and computation resources levels. Some proactive protocols are DSDV (Destination-Sequenced-Distance-Vector Routing Protocol, Perkins & Watson, 1994) and WRP (Wireless Routing Protocol, Murthy & Garcia-Luna-Aceves, 1995). They differ by the number of the routing tables, table content, and methods used to send the table updates between the nodes. Some reactive routing protocols are AODV (Ad Hoc-On Demand-Distance-Vector Routing, Das, Perkins & Royer, 2002) and DSR (Dynamic Source Routing, Johnson & Maltz, 2002). This protocols are source routing-based, where each packet knows the entire route to be followed or hop by hop-based, where each packet knows only the next jump in the net. As expected, some hybrid approaches were proposed to improve the performance and decrease disadvantages. The result seems to have a better scalability that their ancestors because the near nodes collaborate so the overload of route discovering is reduced. Typical examples are ZRP (Zone Routing Protocol, Hass & Pearlman, 1999) and DDR (Distributed Dynamic Routing, Nikaein, Laboid & Bonnet, 2000). Each type of protocols can use a plain or hierarchic routing approach.

3. The Model of the Distributed Framework

The mobile agents framework provides the end-user with the possibility to easily build a virtual cluster that fits its needs and also to find and use existing services. This virtual cluster guarantees to contain only the nodes that are capable of overall performance increase. The framework was entirely developed using the Java programming language. It provides support for developing any kind of agent, such as load balancing agents, security agents, communication agents, mobile or stationary agents, intelligent agents or state agents and so on. As for the communication protocol, the framework uses Remote Method Invocation, RMI.

In order to create a feasible infrastructure of agents, we must first consider some basic models for the entities involved in the system (Zaharia & Leon, 2006; Zaharia, Leon, Calistru & Gâlea, 2008).

3.1. The Task Model

In the proposed model a task is defined as a quadruple (d_c, d_r, t_m, c_m), where:

- d_c: request size (characteristics of the code to be solved);
- d_r: refers to the estimated size of the result. This is needed to compute the returning costs. Of course this is an estimation because there are situations when the user cannot accurately predict this factor (e.g. PSpice simulations);

- t_m: is the maximum time allowed for the task to be executed. This is required because some code error can drive to infinite loops, or the task has an undesired high complexity design. The latter situation can lead to unjustified higher loads at runtime especially for the NP classes of problems. As the result t_m was conceived as an estimation of maximum execution time for the task (if the code is correct) over a computing node from the lowest speed class. As a result, uniformity in task handling is ensured no matter what the computing nodes power is. If the task crosses over this time limit, the task will be terminated, and the owner will be notified;
- c_m: represents the maximum cost that a user can afford so as to execute the task. The agents will use this cost to place the tasks on the servers. Those with a higher cost available will be placed on powerful nodes that have higher costs, and the others will be placed only on lower computing power but cheaper nodes.

3.2. The Server Model

In this approach we take into account the IP number of the server and also we associate three different queues for task receiving and manipulation for each server as follows:

- *An entry queue:* receives the task to be solved that comes from users. The user injects a new task in the system on a server of his/her choice. The task is entered into the entry queue of the server;
- *An execution queue:* stores the tasks selected, using cost analyses, to be executed on the server;
- *An output queue:* stores the results that must be returned to the users.

Because in heterogeneous distributed systems the computing node characteristics may vary throughout a large domain, we considered five performance classes. The MIPS factor was used as the parameter to scale servers into domains. As a result, we have the following categories:

- Server with a speed around 1000 MIPS - VS_1, that is equivalent to a node with a Pentium II processor with 600 MHz working frequency;
- Server with a speed around 2000 MIPS - VS_2, that is equivalent to a node with an AMD Athlon processor with 900 MHz working frequency;
- Server with a speed around 3000 MIPS - VS_3, that is equivalent to a node with a Pentium III processor with 1.13 GHz working frequency;
- Server with a speed around 4000 MIPS - VS_4, that is equivalent to a node with a Pentium IV processor with 2 GHz working frequency;
- Server with a speed around 5000 MIPS - VS_5 that is equivalent to a node with a Pentium IV processor with 2.53 GHz working frequency.

Of course, these domains can be modified in accordance with the newest computer architecture available on the market but this is not relevant to the way that agents will interact; therefore no changes are needed for the analysis.

In fact only the price of the resource and the number of MIPS must be changed, because this computational price is related to the computer power.

As a result, we consider the following price classes:

- Price class S_1, related to speed category VS_1, around 2.7 ¢/s;
- Price class S_2, related to speed category VS_2, around 9 ¢/s (≈ 4.5 ¢/s/computational unit);
- Price class S_3, related to speed category VS_3, around 24 ¢/s (≈ 8 ¢/s/computational unit);
- Price class S_4, related to speed category VS_4, around 48 ¢/s (≈ 12 ¢/s/computational unit);
- Price class S_5, related to speed category VS_5, around 85 ¢/s (≈ 17 ¢/s/computational unit).

The prices were chosen using a nonlinear manner because that conforms to an economic approach. An older machine has already reached its amortisement rates and there will also be fewer requests to be used because most of the users need more and more additional computer power in the shortest possible time.

It is important to note that these prices are only some general recommendations, and processors within the

same speed class can have different prices. It is the agents' responsibility in this case to find the cheapest server, whenever possible.

Most of the available machines on the Internet have operating systems capable of measuring their own loads of all the resources. As a result, the agent framework will directly ask this information in order to fulfil agent requirements. Due to the fact that we cannot predict which servers will be used in the dynamic cluster created by the agents to solve one request, it will not be optimal for the servers to share information about their states because this will mean all-to-all continuous communications.

3.3 The Connection Model

Another aspect of the heterogeneous distributed systems is related to various available broadband at different communication levels. In this approach we used only three types of network connections:

- Connection class VC_1, with a 10 Mb/s broadband;
- Connection class VC_2, with a 100 Mb/s broadband;
- Connection class VC_3, with a 3200 Mb/s broadband.

Due to the discrete and geographically distributed nature of heterogeneous distributed systems we must also analyze the inter-node connection cost. In developed societies these costs begin to be insignificant over small and medium clusters. However, these costs must be used in the formula of task execution costs especially when the task is time consuming and the amount of returned data is higher. For our model we consider the following cost domains:

- Price category PC_1, corresponding to speed class VC1, around 1 ¢/MB;
- Price category PC_2, corresponding to speed class VC2, around 1.5 ¢/MB;
- Price category PC_3, corresponding to speed class VC3, around 5 ¢/MB.

As a result, only the information about local load, speed price class, and communication price class is the most interesting about any node in the network.

4. The Emergent Routing Algorithm

Most routing solutions are dedicated to obtaining a better performance. Unfortunately, if one takes into account other criteria, such as the communicational price, additional constraints need to be considered, and therefore several opposite criteria need to be optimized simultaneously. This situation can be encountered especially in the case of mobile computing, when a system can be simultaneously connected to more than one network. Therefore, the routing decision cannot be made only by a mono-criterial analysis. Specific algorithms such as Mobile IP (Perkins, 2002; Johnson, Perkins & Arkko, 2004) are based on the idea of finding the "way home", i.e. wherever he is, the user can connect first to the initial provider, and solve the problems from there. If we analyse this method through the associated costs, one may want to avoid this method, by using similar entities in the neighbourhood.

In our present approach, routing was obtained in an implicit, not explicit, fashion by the use of a dedicated algorithm, as the result of the aggregation of the local behaviours of individual agents. In order to do that each server will maintain a list of agent probabilities to be moved from a server to its neighbours. This list has the means of strengthening or weakening its values based on reinforcement learning, and are therefore dynamically modified on each transport. Those values exist only for active neighbours. If a server is down its transport probability automatically become zero for all neighbours. If a new server enters the network, its transport probabilities are automatically initialized by its neighbours with random little values which will gradually increase with the time and the use of its services.

The main idea in transport probabilities updates is depicted in figure 1. When an agent follows a path *A-B-C* and passes from server *C* on server *D*, at each transport it memorizes the required time to be transported on the next server. On *D* server it will update the reinforcement values that are equivalent to transport probabilities corresponding to previously visited servers. This procedure is based on the fact that a connection is considered symmetric: the time needed to transport from server *A* to server *B* is equal to the time needed to transport the agent from server *B* to server *A*.

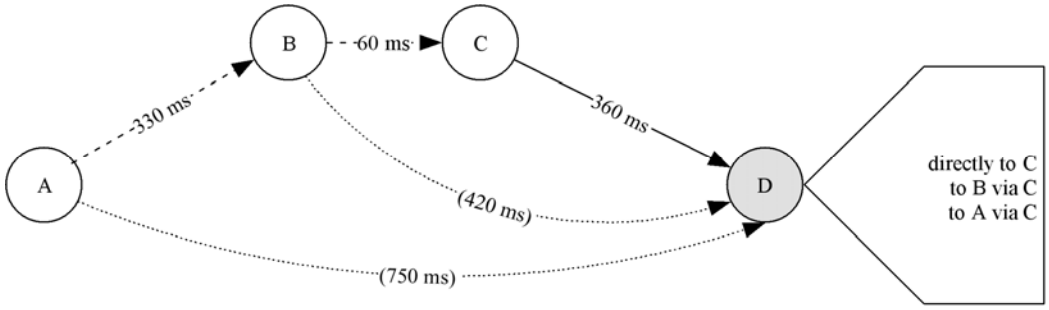

Figure 1. The list of reinforcement values of a server

To compute the reward, $r \in [0,1]$, the following equation is used:

$$r = \frac{MinimumTransportTime}{dt}, \tag{1}$$

where *MinumumTransportTime* is a system constant and *dt* is the time required by an agent to move between two servers. In the implementation of the proposed system a *MinumumTransportTime* of $2.8 \cdot 10^{-6}$ s was considered.

The equation for updating the probabilities is derived from the reinforcement learning paradigm, as follows:

$$P = P + r \cdot (1 - P). \tag{2}$$

In figure 1, the updated rewards on server *D* are presented in the following formulas:

$$r_{D-C} = \frac{2.8 \cdot 10^{-6}}{0.36} = 7.78 \cdot 10^{-6} \tag{3}$$

$$r_{D-C-B} = \frac{2.8 \cdot 10^{-6}}{0.42} = 6.67 \cdot 10^{-6} \tag{4}$$

$$r_{D-C-A} = \frac{2.8 \cdot 10^{-6}}{0.75} = 3.73 \cdot 10^{-6} \tag{5}$$

For example, the probabilities from the reinforcement list of server *C* will be updated by the agents that come back from *D* or by the agents sent from it.

To analyze the routing process using the probabilities of the reinforcement list, let us consider an alternative connection between servers *A* and *D*, as presented in figure 2.

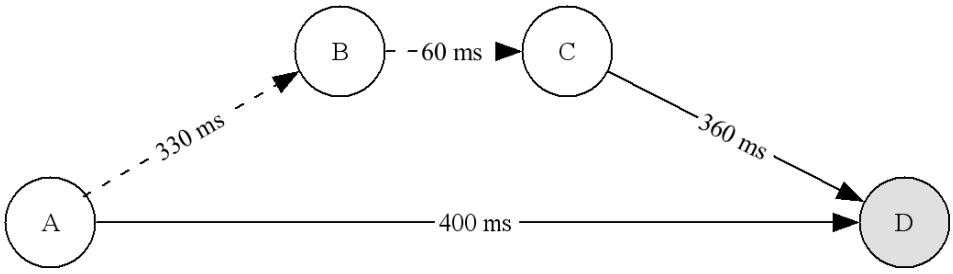

Figure 2. Alternative routes

If an agent comes using this connection, it will receive the following reward:

$$r_{D-A} = \frac{2.8 \cdot 10^{-6}}{0.4} = 7 \cdot 10^{-6} \tag{6}$$

Let us consider the simplest case when only two agents reach server *D*, one of them using the direct route and the other using the alternative route. We assume that the initial probabilities on *D* are zero. In order to make a decision to move from *D* to *A*, if we normalize the probabilities, we will have the situation presented in figure 3. The decision is stochastically made with determined probabilities, and thus the agent will use the direct route with a probability of 65.24% or will follow the alternative route, through *C*, with a probability of 34.76%.

As more agents will continue to come and follow the direct route, the associated probability will increase, therefore the agents from server *D* will choose the direct and quickest route increasingly often.

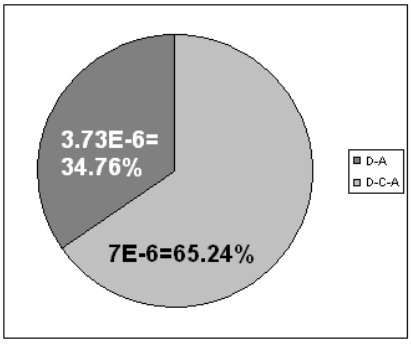

Figure 3. Route selection probabilities

5. Case Study

We will consider a scenario that consists of a system with 3 different servers, 5 initial agents, and 10 tasks to be deployed and executed in the system. Its structure is presented in figure 4. Servers are represented using ellipsoids that contain information about naming, IP, speed, and cost. The connections are arcs, and the associated speed and costs for the communication channels are displayed within the rectangles.

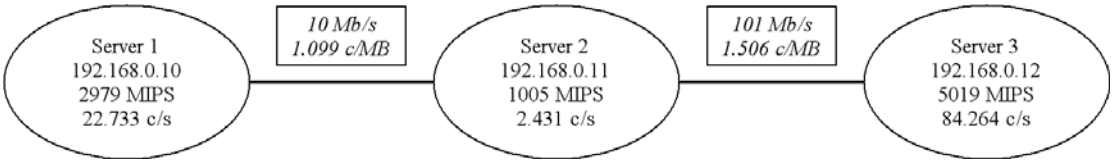

Figure 4. The structure of the distributed system

Below we show the way agents execute in this particular configuration.

Configuration

Servers
Server 1: IP=192.168.0.10 Speed=2979 MIPS Cost=22.733 c/s
Server 2: IP=192.168.0.11 Speed=1005 MIPS Cost=2.431 c/s
Server 3: IP=192.168.0.12 Speed=5019 MIPS Cost=84.264 c/s

Agents
Agents 1 starts on 192.168.0.10
Agents 2 starts on 192.168.0.10
Agents 3 starts on 192.168.0.11
Agents 4 starts on 192.168.0.11
Agents 5 starts on 192.168.0.12

Connections
Connection 192.168.0.10 - 192.168.0.11: Speed=10 Mb/s Cost=1.099 c/MB
Connection 192.168.0.11 - 192.168.0.12: Speed=101 Mb/s Cost=1.506 c/MB

Starting framework

Running agent 2 on server 192.168.0.11
Agent 2 takes task 9 (1854KB, 220KB, 204s, 34.88$)

Running agent 3 on server 192.168.0.11

Running agent 4 on server 192.168.0.12
Agent 4 takes task 0 (354KB, 227KB, 7s, 0.21$)

Running agent 1 on server 192.168.0.10
Agent 1 takes task 1 (396KB, 247KB, 78s, 2.41$)

Running agent 0 on server 192.168.0.10
Agent 0 takes task 3 (1082KB, 274KB, 159s, 19.23$)

Running agent 1 on server 192.168.0.10
Agent 1 waiting for peer answers on server 192.168.0.10

Running agent 0 on server 192.168.0.10
Agent 0 waiting for peer answers on server 192.168.0.10

Running agent 4 on server 192.168.0.12
Agent 4 waiting for peer answers on server 192.168.0.12

Running agent 3 on server 192.168.0.11
Agent 3 moving randomly to server 192.168.0.12

Running agent 2 on server 192.168.0.11
Agent 2 waiting for peer answers on server 192.168.0.11

Running agent 4 on server 192.168.0.12
Agent 4 tries to solve task 0 (354KB, 227KB, 7s, 0.21$) on 192.168.0.11
Task 0 (354KB, 227KB, 7s, 0.21$) to be sent from 2 to 1 by 1

Running agent 3 on server 192.168.0.12
Agent 3 waiting for scouts on server 192.168.0.12

Running agent 1 on server 192.168.0.10
Agent 1 tries to solve task 1 (396KB, 247KB, 78s, 2.41$) on 192.168.0.11
Task 1 (396KB, 247KB, 78s, 2.41$) to be sent from 0 to 1 by 1

Running agent 2 on server 192.168.0.11
Agent 2 tries to solve task 9 (1854KB, 220KB, 204s, 34.88$) on 192.168.0.12
Task 9 (1854KB, 220KB, 204s, 34.88$) to be sent from 1 to 2 by 2
Server 192.168.0.10 begins solving task 3 (1082KB, 274KB, 159s, 19.23$)
Link 1-2 sending task 0 (354KB, 227KB, 7s, 0.21$) to server 192.168.0.11. Cost 0.23c Time 62.954ms
Task 0 (354KB, 227KB, 7s, 0.21$) to be solved on server 192.168.0.11
Link 1-2 sending task 9 (1854KB, 220KB, 204s, 34.88$) to server 192.168.0.12. Cost 1.20c Time 329.709ms
Task 9 (1854KB, 220KB, 204s, 34.88$) to be solved on server 192.168.0.12
Link 0-1 sending task 1 (396KB, 247KB, 78s, 2.41$) to server 192.168.0.11. Cost 0.35c Time 359.610ms
Task 1 (396KB, 247KB, 78s, 2.41$) to be solved on server 192.168.0.11

Running agent 2 on server 192.168.0.10
Agent 2 takes task 8 (133KB, 288KB, 35s, 4.23$)
Agent 2 tries to solve task 8 (133KB, 288KB, 35s, 4.23$) on 192.168.0.12
Task 8 (133KB, 288KB, 35s, 4.23$) to be sent from 0 to 2 by 1
Server 192.168.0.11 continues solving task 7 (906KB, 1370KB, 113s, 3.50$)
Server 192.168.0.12 continues solving task 4 (1878KB, 358KB, 252s, 43.09$)

Server 192.168.0.12: task 4 (1878KB, 358KB, 252s, 43.09$) was successfully solved. Cost 29.74$ / 43.09$
Link 0-1 sending task 8 (133KB, 288KB, 35s, 4.23$) to server 192.168.0.11. Cost 0.26c Time 261.535ms
Task 8 (133KB, 288KB, 35s, 4.23$) to be sent from 1 to 2 by 2
Link 1-2 sending task 8 (133KB, 288KB, 35s, 4.23$) to server 192.168.0.12. Cost 0.19c Time 25.608ms
Task 8 (133KB, 288KB, 35s, 4.23$) returned home on server 192.168.0.12

Running agent 4 on server 192.168.0.10
Agent 4 waiting for scouts on server 192.168.0.10

Running agent 3 on server 192.168.0.10
Agent 3 moving randomly to server 192.168.0.11

Running agent 0 on server 192.168.0.10
Agent 0 moving randomly to server 192.168.0.11
Link 0-1 sending task 6 (390KB, 253KB, 174s, 5.39$) to server 192.168.0.10. Cost 0.22c Time 229.751ms
Task 6 (390KB, 253KB, 174s, 5.39$) returned home on server 192.168.0.10

Running agent 3 on server 192.168.0.12

Running agent 1 on server 192.168.0.11
Agent 1 moving randomly to server 192.168.0.10

Running agent 4 on server 192.168.0.10
Agent 4 moving randomly to server 192.168.0.11

Running agent 2 on server 192.168.0.11

Running agent 0 on server 192.168.0.11
Report: Server 192.168.0.10: task 6 (390KB, 253KB, 174s, 5.39$) was successfully solved. Cost 3.26$ / 5.39$

Finished

The statistics referring to task execution are presented in table 1.

Table 1. Task execution

Task ID	Origin server	Executing server
0	2	1
9	1	2
3	0	0
1	0	1
2	2	1
8	2	0
4	0	2
7	2	1
5	0	1
6	0	1

In table 2 the number of tasks that originate from a server and are executed on other server in the system is presented.

Table 2. Statistics regarding task execution

Origin / Execution	0 (192.168.0.10)	1 (192.168.0.11)	2 (192.168.0.12)
0 (192.168.0.10)	1	3	1
1 (192.168.0.11)	0	0	1
2 (192.168.0.12)	1	3	0

6. Conclusions

In this paper we described a system of autonomous agents, in the sense that each agent decides its actions based on its internal state and the state of the environment, without an explicit external command, other that the task to be executed, which is the actual goal of the agent. However, the autonomy is not incompatible to the global order. The system manifests a form of self-organization, resulting in the routing behaviour of the multiagent system as a whole. The proposed emergent routing algorithm has a dual effect. An agent discovers the routes but also uses the previous information about the routes stored by the previous agents that have already travelled through the local neighbourhood of a server. Since the agent-based approaches are scalable and robust by nature, the research can be continued in the future by considering a larger number of servers and agents, which will allow the study of the real life problems on massive infrastructures, such as the GSM internet servers.

REFERENCES

1. CARRILLO, L., MARZO J. L., VILÀ P., MANTILLA C., **MAntS-Hoc: A Multi-agent Ant-based System for Routing in Mobile Ad Hoc Networks,** http://eia.udg.es/~lilianac/publications/ccia04_Liliana-Final2.pdf, 2004.
2. DAS, S., PERKINS C., ROYER E., **Ad Hoc on Demand Distance Vector (AODV) Routing**, Internet Draft, draft-ietf-manetaodv-11.txt, 2002.
3. HASS, Z. J., PEARLMAN R., **Zone Routing Protocol for Ad-hoc Networks**, IETF MANET Internet Draft, 2002.
4. JOHNSON, D. B., MALTZ D. A., **The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks,** Internet Draft, draft-ietf-manet-dsr-07.txt, 2002.
5. JOHNSON, D., PERKINS C., ARKKO J., *RFC 3775,* **Mobility Support in IPv6,** The Internet Society, http://tools.ietf.org/html/rfc3775, 2004.
6. MARWAHA, S., THAM C. K., SRINIVASAN D., **A Novel Routing Protocol Using Mobile Agents And Reactive Route Discovery For Ad Hoc Wireless Netwokrs**, www.ece.nus.edu.sg/stfpage/eletck/ papers/ShivaICON_conf8a51.pdf, 2005.
7. MURTHY, S., GARCIA-LUNA-ACEVES J. J., **A Routing Protocol for Packet Radio Networks**, ACM/IEEE Proc. of the 1st. Annual International Conference on Mobile Computing and Networking, Berkeley, CA, 1995, pp. 86–95.
8. NIKAEIN, N., LABOID H., BONNET C., **Distributed Dynamic Routing Algorithm (ddr) for Mobile Ad Hoc Networks,** Proceedings of the MobiHOC 2000: 1st. Annual Workshop on Mobile Ad Hoc Networking and Computing, 2000.
9. PERKINS, C. E., WATSON T. J., **Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers**, ACM Proc. of SIGCOMM_94 Conference on Communications Architectures, London, UK, 1994.

10. PERKINS, C., **RFC 3344, IP Mobility Support for IPv4**, The Internet Society, http://tools.ietf.org/html/rfc3344, 2002.

11. WOOLDRIDGE, M., **Intelligent Agents**, in G. Weiß (ed.), Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, Massachusetts, 2000.

12. WOOLDRIDGE, M., JENNINGS N. R., **Intelligent Agents: Theory and Practice**, The Knowledge Engineering Review, Vol. 10(2), 1995, pp. 115-152.

13. ZAHARIA, M. H., LEON F., **System Prototype for the Dynamic Management of Distributed Resources in Mobile Computing Using Intelligent Agents**, CNCSIS project code 66/2005, Technical Report, 2006.

14. ZAHARIA, M. H., LEON F., CALISTRU C. N., GÂLEA D., **Load Balancing in Heterogeneous Distributed Systems Using Mobile Intelligent Agents,** IEEE International Conference on Distributed Human-Machine Sysems, Athens, Greece, 2008.