# Towards Autonomous Mobile Agents with Emergent Migration Behaviour

Tino Schlegel*
Swinburne University of Technology
Faculty of Information and Communication Technologies
Hawthorn, 3122 Victoria, Australia
tschlegel@ict.swin.edu.au

Peter Braun
Swinburne University of Technology
Faculty of Information and Communication Technologies
Hawthorn, 3122 Victoria, Australia
pbraun@ict.swin.edu.au

Ryszard Kowalczyk
Swinburne University of Technology
Faculty of Information and Communication Technologies
Hawthorn, 3122 Victoria, Australia
rkowalczyk@ict.swin.edu.au

## ABSTRACT

Along with manifold advantages of distributed multi-agent systems, increased network traffic produced by highly communicative agents in large distributed systems has to be considered as their practical downside. We suggest to apply the programming paradigm of mobile agents to reduce this network overhead by allowing agents to meet at the same network node before commencing communication. A remote communication between two agents could then be replaced by one or two agent migrations, followed by local communication. Since only in trivial cases it is possible to decide at design time whether remote communication or agent migration with subsequent local communication would perform better, this decision has to be made at run-time based on environmental parameters and agents' past experience. We present an adaptive approach, which is inspired by a solution of the *El Farol* problem. Every agent forecasts the network load of the next communication step and applies a simple mathematical model to decide between the two alternatives at run-time. In addition, our approach does not only consider network load but also server load by enabling agents to dynamically forecast the number of agents migrating to a specific agent server. The approach is evaluated with simulation experiments in static and dynamic server load environments.

## Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents

## General Terms

Performance

*PhD student.


*AAMAS'06* May 8–12 2006, Hakodate, Hokkaido, Japan.


## Keywords

mobile agents, collective and emergent agent behaviour

## 1. INTRODUCTION

In most distributed multi-agent systems, communication and coordination between agents is based on asynchronous message passing. If agents are located on distant agent servers, sending messages between agents generates network traffic between the corresponding network nodes. As every network traffic causes higher execution time by network transmission and latency compared with a centralized solution, a trade-off between advantages of agent distribution and drawbacks regarding their execution time has to be considered. For example, a promising application area of multi-agent systems is the management of Grid services [8, 13, 14]. Agents can act as representative of a service instance and supplement the service's functionality by intelligent techniques for negotiating QoS parameters [12]. Multi-step negotiation based on the iterative contract-net protocol between widely distributed agents can impose high network overhead, which might question the applicability of negotiations at large. We believe that this important problem has not sufficiently been recognized by the agent community so far and current agent development methodologies do not provide solutions to this problem.

Mobile agents have been introduced as a paradigm for distributed systems, in particular to reduce network traffic by moving code close to the place where the data is. A mobile agent is a program that can migrate from one host to another host in a network of heterogeneous computer systems and fulfil a user-given task. It can autonomously work and communicate with other agents and host systems. During the self-initiated migration, the agent carries its code and some kind of execution state with it [4, 21].

Mobile agents are a solution of our scenario of negotiating agents, because both agents could decide to meet at the same agent server and then communicate only locally without generating any network traffic except of the migration. Mobile agents can not only reduce network load and increase application response time but can also improve reliability by making the application more robust against network failures and reduce power consumption in case of mobile devices. In the last years, many research groups have carried out empirical evaluations to evaluate which of these two paradigms

(remote communication or migration followed by local communication) performs best regarding network load and processing time in various application scenarios [15, 17, 18].

It is obvious to see that mobile agents can only produce lower network load, if, simply speaking, the mobile agent's code and state that have to be transmitted are not larger than the amount of data that can be saved by the use of a mobile agent. It depends on the message size, the code and state size, and the network environment to decide whether mobile agents perform better than remote communication. In the following, we call this the *migration decision problem*. Most current solutions of the migration decision problem are based on mathematical models of the application's network load using parameter estimations (e.g. the number of negotiation steps, average message size, etc.) and lead to a decision at design time. We believe that in most application scenarios it will be beneficial to address the migration decision problem at run-time, because only then all influencing parameters (network throughput, latency, request message size, code size) can be actually known.

In this paper we present an approach to solve the migration decision problem at run-time involving dynamic prediction of message sizes based on historical information about previous communication acts. The agents do not only forecast message sizes but also the load of the remote agent server that an agent would migrate to. Although an agent might opt for migration when considering only message sizes, a remote communication could be the better alternative if also server load is taken into account. Our solution is inspired by the concept of inductive reasoning and bounded rationality introduced by Arthur [1] and does not rely on additional communication between the agents to decide which agent should migrate to which agent server. It performs well for large and highly dynamic systems and can easily be adapted to various system sizes. In addition, the overall system performance is not affected in case of agents fail or leave the system.

The remainder of the paper is structured as follows: The next section gives an overview of the related work already done in the area of network aware programs. Section 3 describes the model for communication and coordination costs that is used for this work and our assumptions and constraints. It is following a section describing our simulations and the experimental validation of the simulation results. An outlook to the future work concludes the paper.

## 2. RELATED WORK

Current approaches for deciding between remote communication and agent migration are based on static mathematical analysis of the application and network models. We argue that such static approaches are not sufficient for all kinds of application and should be complemented by adaptive decision models which enable the agent to decide between remote communication and migration during run-time autonomously.

To illustrate our discussion, we use the following simple model. In the remote communication approach, agent $A_1$ sends a request message of size $B_{req}$ to the remote agent $A_2$, which answers with a result message of size $B_{res}$. The amount of bytes that are sent over the network equals $B_{RC} = B_{req} + B_{res}$. In the migration approach, agent $A_1$, which contains code of size $B_c$ and state information of size $B_s + B_{req}$ migrates to the remote agent server (here we confine

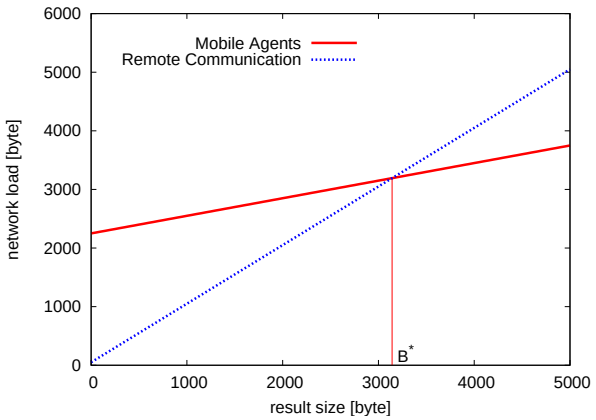


**Figure 1: Evaluation of a simple mathematical model with [kB]** $B_{req} = 50, B_c = 2000, B_s = 100, \varsigma = 0.7$ **and break-even point** $B^* \approx 3143$

to one mobile agent, instead of two as in the more general case mentioned in the previous section). At the remote agent server, the agent communicates locally with agent $A_2$, which does not produce any network load. Agent $A_1$ has code to filter or compress the reply message, so that only $(1-\varsigma)B_{res}$ must be carried to the agent's home agency. Note that the agent neither carries its code nor the request message back home, because the code is already available at the home server and the request message is no longer needed. However, state information must be sent back to the agent's home agency. Thus, the amount of bytes sent in the mobile agent approach equals $B_{MA} = B_c + 2B_s + B_{req} + (1-\varsigma)B_{res}$. When we evaluate this model using an artificial parameter setting, we see that there is a break-even point $B^*$, at which the migration overhead produced by mobile agents' code and state relocation is compensated by their ability for data reduction (compare Fig. 1).

Picco's approach is to estimate all influencing parameters and decide statically between these two design paradigms [16]. That approach is suitable for scenarios in which all the parameters are known to have constant size (e.g. the result message is a telephone number). Even if the size of parameters (e.g. the result size) are Poisson distributed with an expectation $\lambda$ and $B^* \gg \lambda$ or $B^* \ll \lambda$ (i.e. the probability for a wrong decision is very low), a static decision is sufficient. In contrast, if $B^*$ is close to $\lambda$ or the distribution of reply sizes is not Poisson, changes over time or is simply unknown, a static decision on design paradigms is error-prone.

Another drawback of Picco's approach is that it does not take into account the possibility of a mixture of remote communications and agent migrations. It was already identified by Chia and Kannapan to be beneficial in terms of network load in the case of only two data servers [7]. Later, Strasser and Schwehm developed an algorithm to determine the optimal agent's itinerary of $n$ data servers under the assumption of full knowledge [19]. In their approach, the agent only migrates to a subset of all data servers, whereas the other servers are accessed using remote procedure calls. The optimal agent itinerary depends on the size of requests and results, the number of communication steps and on the network quality between each pair of nodes. All these parameters are assumed to be estimated or known in advance.

Later, this approach was extended by a technique to provide information about network quality using distance maps before planning the optimal migration itinerary [20].

The second problem that has to be considered is the load of the agent server to which a mobile agent wants to migrate to. Even if a migration is beneficial from the perspective of network load, a migration has to be dismissed, if the destination agent server is already overloaded by other agents or server tasks. This problem has mainly been accounted from the server or system perspective so far. For example, Flüs [9] has proposed techniques for capacity planning of mobile agent servers and extended the Tracy [3] agent toolkit by techniques to avoid server overloading by refusing or queuing incoming agents. From a system perspective, the problem of load balancing between different agent servers is very important. Georgousopoulos et al. [10] present a load balancing mechanism based on a combination of state and model-based approaches. Cao et al. [5] describe a load balancing mechanism for Web servers based on the mobile agent paradigm.

A combined solution of both problems can be considered as a self-organizing approach to minimize network and server load of a multi-agent systems. To the best of our knowledge, there is no approach in the literature that combines both network load optimization and server load optimization so far.

## 3. MODEL DESCRIPTION

We model a distributed multi-agent system as a network of nodes $\mathcal{L} = \{L_1, \ldots, L_\mathrm{m}\}$ on each of which an agent server is running. The agent system comprises of agents $\mathcal{A} = \{A_1, \ldots, A_\mathrm{n}\}$, each located on its home agent server $h(A_i)$ at the beginning of its life-cycle. The map $\mathcal{P} : \mathcal{A} \rightarrow \mathcal{L}$ defines the location of each agent in general. Agents that are bound to large databases or otherwise immobile legacy systems are called stationary. All other agents are considered to be mobile.

Each agent has to process a list of communication steps $\mathcal{C} = \{c_\mathrm{i} \mid i : 1 \ldots p\}$ as part of its task. Each communication step $c_\mathrm{i} = \langle A_\mathrm{a}, A_\mathrm{b}, m_\mathrm{k}, m_\mathrm{j} \rangle_i$ defines that a request message $m_\mathrm{k}$ is sent from agent $A_\mathrm{a}$ (source) to agent $A_\mathrm{b}$ (destination), which responses with a reply message $m_\mathrm{j}$. Each message $m_\mathrm{k}$ can be seen as an arbitrary sequence of bytes of the length $B_\mathrm{m}(m_\mathrm{k})$. The network cost for a remote communication step is calculated as follows:

$$B_{RC}(c_\mathrm{i}) = \begin{cases} 0 & if\ \mathcal{P}(A_\mathrm{a}) = \mathcal{P}(A_\mathrm{b}) \\ B_\mathrm{m}(m_\mathrm{k}) + B_\mathrm{m}(m_\mathrm{j}) & otherwise \end{cases} \quad (1)$$

Before each communication step, both involved agents (provided they are mobile) have to decide, whether to migrate from their current location $\mathcal{P}(A_\mathrm{i})$ to another location $\mathcal{P}(A_\mathrm{i})'$ before commencing the communication act. In the following we assume that both agents migrate to the same destination server. The network load only comprises of the cost for transferring the agents' code of size $B_c$ and state of size $B_s$ (which is equal to the size of the request message) but no costs for local communication. After the communication, both agents migrate back to their home agent servers (without code, because the code is already available at their home agent servers). Agent $A_\mathrm{a}$ carries back state information and results of size $(1-\varsigma)B_\mathrm{m}(m_\mathrm{j})$ with $0 \leq \varsigma \leq 1$, because the size of the original reply message could be reduced (e.g. filtered or compressed). The mobile agent communication network cost is calculated by

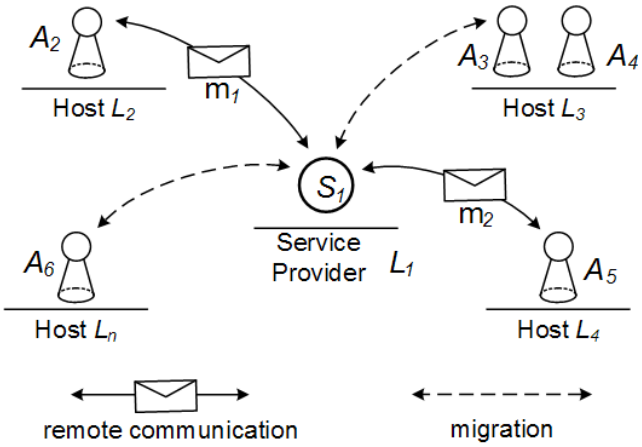


**Figure 2: Communication model with multiple agents and a single server.**

$$B_{MA}(c_\mathrm{i}) = \begin{cases} 0 & if \mathcal{P}(A_\mathrm{a}) = \mathcal{P}(A_\mathrm{b}) \\ B_c(A_\mathrm{a}) + B_\mathrm{m}(m_\mathrm{k}) + & \\ B_c(A_\mathrm{b}) + (1-\varsigma)B_\mathrm{m}(m_\mathrm{j}) & otherwise \end{cases} \quad (2)$$

if both agents are mobile. If only the sender agent is mobile, code transmission of agent $A_\mathrm{b}$ has to be disregarded.

The last formula is independent of the location of the destination agent server because we only consider network load and not transmission time at this stage. Formulas 1 and 2 can be used to decide between the two alternatives at run-time, provided that all message sizes are known. To further reduce the complexity of our model, we assume for our first experiments that all agents exchange messages with only one agent $S_1$, which is stationary and located on $L_1$, see Fig. 2.

## 4. TOWARDS EMERGENT MIGRATION BEHAVIOUR

### 4.1 Introduction – The El Farol Problem

We present a combined approach for reducing network load and agent server load, which is based on predicted environment parameters at run-time. As mentioned in the previous section, an agent has to decide before each communication step $c_\mathrm{i}$ between remote communication and agent migration. This decision must be based on limited knowledge, which only includes its own code size $B_c$ and the request message size $B_\mathrm{m}(m_\mathrm{k})$. The size of the reply message $B_\mathrm{m}(m_\mathrm{j})$, the selectivity factor $\varsigma$, and the current remote agent server load are not known in advance.

Naturally, we have to assume that neither of these values is distributed uniformly within their domain. In that case, there would be no chance to predict these values based on past experience. In order to apply machine learning algorithms for these values, they must render a certain pattern, for example some kind of probability distribution. One of our hypotheses is that in many application scenarios the reply size can be described by well-known distributions such as

Poisson or Gaussian distributions. It should be clear that this approach can be successfully applied to many application scenarios, for which a static analysis is possible as proposed by Picco [16] and Schwehm and Strasser [19]. In these cases, the reply message size, selectivity factor, and server load have a straightforward distribution (e.g. a constant value), which is easy to learn by our approach.

The prediction mechanism is inspired by inductive reasoning and bounded rationality principles introduced by Arthur [1] and demonstrated as solution of the El Farol bar problem. The original problem setting consists of 100 people to decide repetitively and independently, whether to go to the El Farol bar on a specific day of the week or not. All people have the same preference in order to decide: they will go to the bar, only if they expect less than 60 people to show up. Choices are unaffected from previous visits. There is no communication between people (e.g. to form coalitions or coordinate their decisions) and the only information available to every agent is the number of people that attended the bar in the past weeks.

Arthur's solution of the El Farol bar problem consists of a set of predictors for each patron, each able to forecast the number of patrons by applying very simple pattern based on past experience. At the beginning of the simulation, each person is assigned a randomly chosen set of predictors. In each step of the simulation the predictor that has performed best in the last week, is used to forecast the next number of attendants. The interesting result of Arthur's simulation shows a fast stabilizing process around the optimal number of 60 attendants. For more information about the El Farol bar problem, we refer to [6].

Our problem can be mapped to the El Farol bar problem by assuming agent $S_1$ (to which all communication is directed to) and its agent server to be the bar with limited resources. All other agents have to decide whether to migrate to that agent server or not. An agent will migrate only if the migration will reduce network costs and the destination agent server is not overloaded yet. The latter requirement can be simply modelled by a number of agents executed in parallel. We have to deal with similar effects as in the El Farol problem: if all agents believe that the server will not be overloaded, all agents will migrate and if all agents believe that the server will be overloaded, no agent will migrate – both experiences would invalidate the agents' beliefs in the environment.

The advantage of our approach is that no additional communication overhead is introduced because agents learn from their past experiences. In contrast to the original El Farol problem, we do not assume free information dissemination about the current server load to agents that did not migrate to the agent server recently. That is, only agents that have migrated to the agent server recently can benefit from up-to-date knowledge.

## 4.2 Network Cost Prediction

Each agent predicts environment parameters that are not know in advance but necessary for the migration decision based on its own historical data. For the prediction of message sizes and selectivity factors, this technique provides good results because we do not only use one predictor but a set of different predictors to provide a better accuracy.

The predictor that is used for the next prediction of the message size and the selectivity factor is called active predictor. After an agent has executed a communication step, its history information is updated with the measured values and all predictors will be rated again. The most accurate predictor in the last steps is chosen as the new active predictor because it is the most prospective for the next prediction. To reduce the computational overhead of the predictor evaluation, it is possible, to use the active predictor until it's relative error $\delta$ exceeds a pre-defined threshold.

As the migration decision of agent $A_{\mathrm{j}}$ based on the network costs is independent of any other agent $A_{\mathrm{k}}$, it is fair to assign all agents the same set of predictors. Of course, the quality of the predictors influences the quality of the predicted values and it might be reasonable to introduce specialized predictors that can be applied to specific probability distributions. However, in our first experiments we were keen to learn about the quality of very simple predictors. Actually, it is not important to achieve a *correct prediction* of the environment parameters rather than making the *correct decision* based on the predicted values. The best predictor will be chosen automatically, while less accurate predictors will not be considered in the future.

For our experiments we have developed the following predictors:

- same value as $n^{th}$last communication step,
- moving average of last $n$ steps,
- the trend of the last $n$ steps,
- random value in the interval between current minimum and maximum,
- Poisson distribution based on the current history data.

## 4.3 Agent Server Load Prediction

Agent server load prediction differs slightly from message size prediction, because all agents compete for the same resource and agents' decisions regarding server load are dependent on each other. If all agents would predict the same resource load, than this would invalidate their beliefs. Therefore, it is important to ensure agents make different and adaptive decisions in their environment.

The selection of the active predictor differs from the technique presented previously. In contrast to the message reply size and selectivity prediction, the selection of the next server load predictor is not deterministic but has a level of uncertainty. The probability that a predictor is chosen as the new active predictor increases with the predictor's accuracy. The accuracy of a predictor is calculated from the number of correct and wrong decisions, i.e. the agent has predicted a not overloaded resource and decided to migrate.

Only the last $n$ values are considered for calculation of the predictors accuracy because this allows a faster adaptation to changes. The number $n$ can be configured by the agent designer. For every correct decision (the agent migrated and the resource was not overloaded), a predictor gets a positive rating. For a wrong prediction only the responsible active predictor gets a negative rating. To improve the learning process for new predictors and predictors with old historical values, they get a positive rating for correct predictions, but no penalty for wrong predictions, until they reach $m$ predictions with $m \leq n$. The overall rating is the sum of all positive and negative ratings. The relative error of the prediction is not considered for the resource load prediction.

For example, if predictor $P_1$ predicted in 60% of the considered cases that the resource is not overloaded and the agent migrates to the resource while predictor $P_2$ predicted only 30% of the cases correct the probability of choosing $P_1$ is twice the probability of choosing $P_2$.

Usually, the agent bases the migration decision on local knowledge which means the resource load information is updated only if the agent migrates to the resource. Some experiments were conducted with global knowledge in which we updated the resource load information after every migration decision to learn about the differences.

The current approach uses a set of simple predictors, similar to the El Farol approach. Developed predictors include:

- $n$-period cycle predictor: predicts the same as $n^{th}$-last history value.
- moving average predictor: predicts the average of all values in a window of the last $n$ history values.
- linear or non-linear predictors: predicts interpolated values the considers the last $n$ history values.

## 4.4 Rule Based Configuration

For the strategic migration decision the agent uses a rule based decision system. The rules are based on the parameters provided by the message size and server load prediction to check which paradigm offers a better performance. Different rule sets can be configured to provide different configurations and priorities. The owner of an agent can decide which of the rule sets from a number of predefined rule sets are to be used for the decision making to meet different user preferences.

### 4.4.1 Data Collecting Phase

In the initial phase when agents have no (or not enough) historical data it is necessary to use a different type of predictors–not based on historical information and not predicting message size, but simply deciding between remote communication and migration, such as:

- Random migration strategy: The agent migrates with a given probability to the destination host.
- Always migrate strategy: The agent always migrates to the destination host. This is the fastest way to get required information for the learning phase.
- Never migrate strategy: The agent always uses remote procedure calls. In this case, the agent will never switch to the learning phase because no resource load information is updated.

After the agent has collected enough data so that at least some of the predictors are working, the agent switches to the working phase.

### 4.4.2 Learning and Working Phase

In the working phase, the agent uses the predicted data of message reply sizes, agent selectivity, and the expected agent server load based on local knowledge to come to a migration decision. Based on the rule set that reflects best the user preferences, an agent makes an autonomous decision which paradigm offers the highest benefit for the user.

- Minimal Network Load: Only the network load is considered for the migration decision. If the resource is crowded, the agent accepts waiting times.
- Balanced strategy: The agent will use migration when it offers benefit compared to RPC and the resource won't be crowded.
- Best Performance: Same as minimal network load, except that the migration probability increases linear with the age of the newest historical data or the average age of historical data in the sliding window, if the resource load data is out of date. This prevents that an agent create the habit not to migrate, if it has decided once like this. The predictor always predicts not to migrate, based on the same historical values.

# 5. EXPERIMENTAL VALIDATION

## 5.1 Outline

We have developed a first simulation test-bed in the Java programming language for a distributed multi-agent system independent of any specific agent toolkit. Our simulation environment enables different configurations of the number of agents, the number of communication steps for each agent, various distributions of reply sizes and selectivity factors for each agent, etc. The simulation uses an event-driven model to trigger agent activities. Network load equals the number of bytes transmitted without taking network protocol overhead (IP, TCP, application layer) into account. The latter will be considered in the near future, when the simulation environment will be re-designed to work with a well-known network simulation tool such as NS-2 or Omnet++. Network transmission time is modelled by taking bandwidth and latency between each pair of network nodes into account.

We have tested a number of different configurations of our simulation environment and for the evaluation we measured the following system metrics:

**Reply size:** The size of the reply message.

**Selectivity factor:** The agent's selectivity factor for the reply message.

**Agent server load:** The number of agents that attended the server in the last simulation step.

Several sets of experiments were conducted with different simulated configurations. We repeated each experiment several times and show results of representative experiments and some average values for illustration.

The first set of experiments focuses on the accuracy of message size and selectivity factor prediction. Both form the foundation for the strategic migration decision based on network load. The second set of experiments focuses on prediction of message size as well as agent server load, and investigates if and how fast agents can adapt to varying levels of server availability.

Usually we run experiments while varying only one parameter and keeping all other parameters constant. A detailed description of the simulation environment and the observed parameters follows in the next sections.

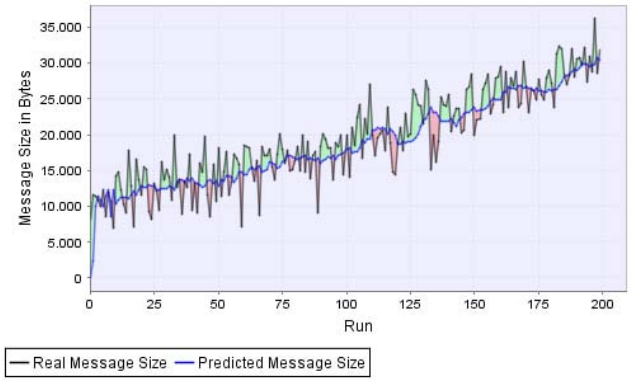


**Figure 3: Comparison of the predicted message size and the real message size.**

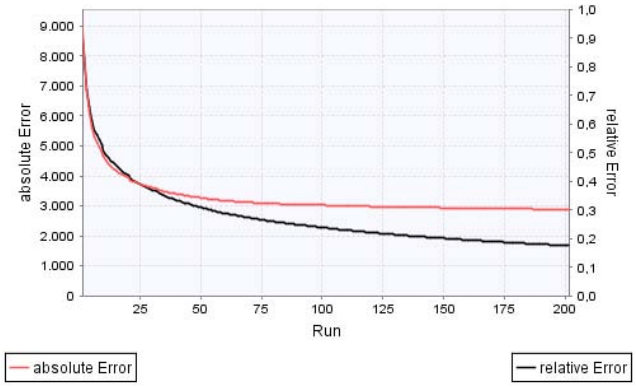


**Figure 4: Average absolute and relative error of the reply message size prediction.**

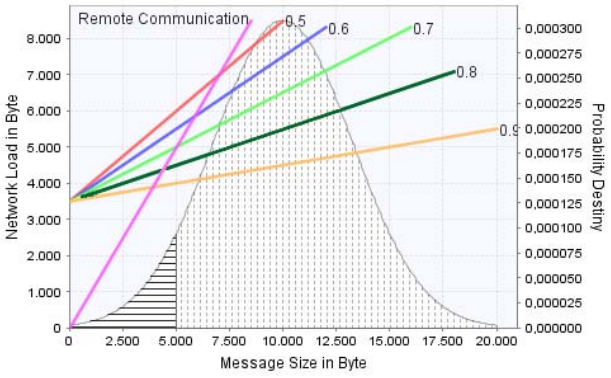


**Figure 5: Comparison of mobile agent and remote communication paradigm.**

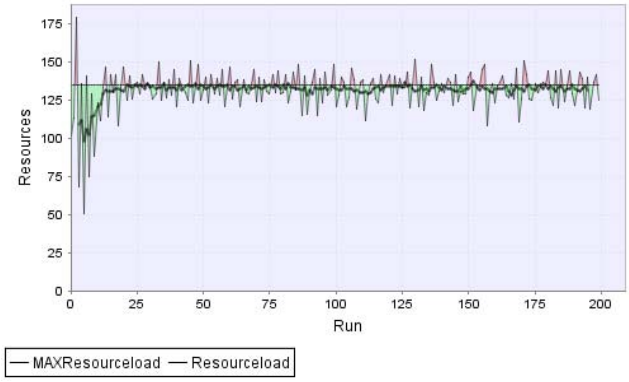


**Figure 6: Development of server load for 200 cycles with resource limit of 135 units.**

## 5.2 Message Size Prediction

In this section we report on results of message size and selectivity factor prediction. For this set of experiments we have selected different artificial Gaussian distributions of the message size with parameters $\mu = 5000$, 10000, 50000 Byte and $\sigma^2 = 10\%$, 20%, 40% of these mean values.

To validate the migration decision, we set the other parameters as follows: the agent size is 35 KByte, the agent performs 200 predictions using a set of 7 different predictors. The predictors for this experiment include 5 different period cycle predictors, a moving average predictor, and a linear predictor. Fig. 3 shows one example of message size prediction, in which the real message size is Gaussian distributed $N(10000Byte, 3250Byte)$ with an increasing mean value. We measured the absolute and relative error of the prediction and prediction correctness, i.e. whether the predicted value leads to the correct migration decision. Fig. 4 shows that the absolute error finally reaches 3000 KB and the relative error is below 20%.

In addition, we measured the cumulative values of these parameters. Figure 5 compares the network load of the remote communication and migration approach for different selectivity factors and message reply sizes for a representative message reply size distribution with $\mu = 10000$ $Byte$ and $\sigma = 32,5\% = 3250$ $Byte$. Considering only a statistical analysis of a fixed set of parameters and the given message reply size distribution, we can calculate that the break even point for the mobile agent paradigm with a selectivity of 70% is at 5000 Byte. An incorrect decision is made in less than 6,3% of all cases. We expect that real applications have certain patterns for the reply size distribution, and a small derivation.

## 5.3 Combined Prediction

In this section we report on results of combined message size and agent server load prediction mechanisms. In the first experiments we have selected a set of 200 agents with same agent resource requirements of 1 unit (memory, processor cycles). These agents want to use agent $S_1$ located on host $L_1$, which has available resources of $n = 135$ units. Fig. 6 shows the agent server load usage when the available server resources are constant. It can be seen that after only few executions, the average resource load balances below the maximum value. Only in few cases, the prediction over-estimates the real number of resources available. Fig. 7 shows that the number of migrations is similar for all agents. There is no group of agents that always migrates while other agents never migrate.

A second set of experiments uses the same parameters as the previous one, but the available server resources are varied at run-time after every 75 executions. The goal of this experiment is to see how fast the agents prediction mechanism can adapt to varying environmental parameter. We

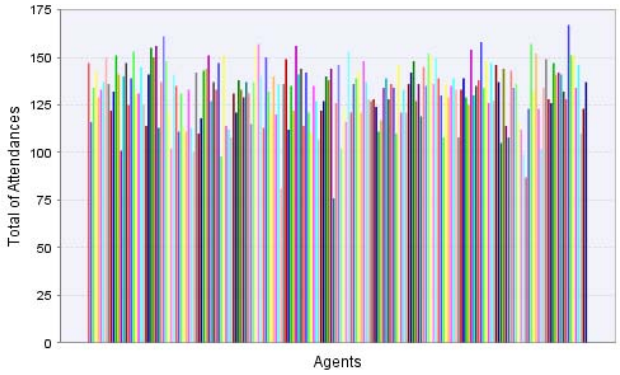


**Figure 7: Attendance of mobile agents at the remote agent server for constant resource limit.**

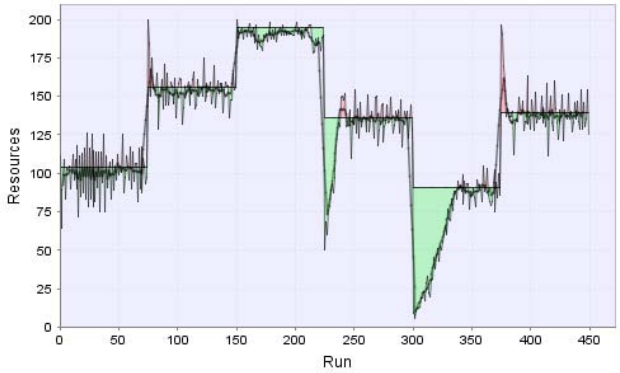


**Figure 8: The adaptation of one server to varying resource capabilities.**

set up an environment with different numbers of agents, and varying resource capabilities over time and measured the number of resources used by agents. We conducted experiments in which all agents have the same resource consumption as well as different resource consumption. Fig. 8 shows for one agent server that the adaption to varying server capabilities works surprisingly fast. Only in few cases, the available server capabilities are over-estimated, whereas in many cases the current server capabilities are close to the maximum values. In addition, Fig. 9 shows that even in the case of varying server resources the number of migrations is similar for all agents.

Finally, we tested two variants of updating the historical data of agents. At first, agents update their history values after every execution step, independent of having migrated to that server or not. This case resembles the original El Farol behaviour. Secondly, agents have limited and heterogeneous agent server load information. History values are only updated if an agent has migrated to the remote agent server.

Interestingly, the adaptive solution works better when agents have only limited and heterogeneous knowledge. Furthermore, the limited knowledge is preferred in real systems because it has no additional communication costs and it would be difficult to distribute resource information to potential agents.

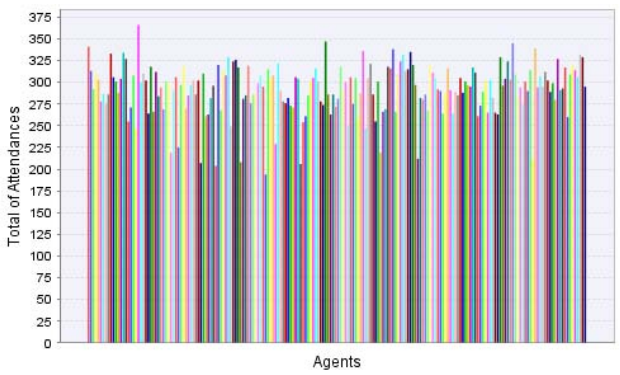


**Figure 9: Attendance of mobile agents at the remote agent server for varying resource capabilities.**

## 6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an approach for reducing communication overhead in a distributed multi-agent system by agent mobility at run-time. Our approach also considers system performance by balancing agent server load. Our approach is based on mobile agents that decide at run-time between remote communication and migration. It was inspired by inductive learning and bounded rationality principles. The prediction of the required environment parameters enables mobile agents to autonomously make a strategic migration decision and adapt to the dynamic environment. From a system perspective, this solution can be considered as a self-organizing mechanism to minimize network load and server load in a multi-agent system.

We have presented and evaluated a number of simulation experiments involving agents in static and dynamic server load environments. The results show that this new approach for strategic migration optimization based on inductive reasoning and bounded rationality principles is very promising and justifies further investigations. The advantages of this approach are its fast adaptability to different system sizes, its simplicity, and its easy extensibility. These are known advantages, although in this particular application area rarely used. With different rule sets, agents can be configured for different user preferences.

A possible improvement would be to share historical data between all agents of one agency in order to increase the learning speed and provide better predictions since historical data are up-to-date. After we have proven that this approach works fine with the event driven model we will upgrade our Java implementation. Our next step is to switch to a more realistic asynchronous simulation environment. The simple event-driven *multiple agents using one resource* approach will be extended with techniques for multiples resources. Agents will not only migrate to one agent server, but can freely migrate in the whole multi-agent system. Each agent will have a set of tasks to fulfil that need a certain amount of system capabilities. A set of system providers located on some of the servers offer services that also consume system capabilities. These services are used by each agent to fulfil its task either using remote communication or agent migration.

Our approach for optimizing network load in multi-agent systems must be supplemented in the longer run with tech-

niques to optimize the migration behaviour of mobile agents. In this paper we have assumed rather simple migration techniques (the agent carries all its code and data information), despite of already known techniques to optimize the migration strategy of mobile agents [2, 11].

## 7. REFERENCES


[1] W. B. Arthur. Inductive Reasoning and Bounded Rationality. *American Economic Review (Papers and Proceedings)*, 84(2):406–411, May 1994.

[2] P. Braun and S. Kern. Towards adaptive migration techniques for mobile agents. In Z. Guessoum, editor, *Fifth Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS 2005), Paris (France), March 2005*, 2005.

[3] P. Braun, I. Müller, T. Schlegel, S. Kern, V. Schau, and W. Rossak. Tracy: An extensible plugin-oriented software architecture for mobile agent toolkits. In M. Calisti, M. Klusch, and R. Unland, editors, *Software Agent-Based Applications, Platforms and Development Kits*, Whitestein Series in Software Agent Technologies, pages 357–382. Birkhäuser Verlag, 2005.

[4] P. Braun and W. R. Rossak. *Mobile Agents–Basic Concept, Mobility Models, and the Tracy Toolkit*. Morgan Kaufmann Publishers, 2005.

[5] J. Cao, Y. Sun, X. Wang, and S. K. Das. Scalable load balancing on distributed web servers using mobile agents. *Journal of Parallel Distributed Computing*, 63(10):996–1005, 2003.

[6] D. Challet, M. Marsili, and G. Ottino. Shedding light on El Farol. Technical Report 0406002, Economics Working Paper Archive at WUSTL, June 2004.

[7] T.-H. Chia and S. Kannapan. Strategically mobile agents. In K. Rothermel and R. Popescu-Zeletin, editors, *Proceedings of the First International Workshop on Mobile Agents (MA'97), Berlin (Germany), April 1997*, volume 1219 of *Lecture Notes in Computer Science*, pages 149–161. Springer-Verlag, 1997.

[8] L. Chunlin and L. Layuan. Apply agent to build Grid service management. *Journal of Network and Computer Applications*, 26:323–340, 2003.

[9] C. Flüs. *Capacity Planning of Mobile Agent Systems Designing Efficient Intranet Applications*. PhD thesis, Universität Duisburg-Essen (Germany), Feb. 2005.

[10] C. Georgousopoulos and O. F. Rana. Combining state and model-based approaches for mobile agent load balancing. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 878–885, New York, NY, USA, 2003. ACM Press.

[11] S. Kern and P. Braun. Towards adaptive migration strategies for mobile agents. In *Second GSFC/IEEE Workshop on Radical Agent Concepts (WRAC), NASA Goddard Space Flight Center, Greenbelt, MD (USA), September 2005*. Springer Verlag, 2005.

[12] A. Ludwig, P. Braun, R. Kowalczyk, and B. Franczyk. A framework for automated negotiation of service level agreements in service grids. In *1st International Workshop on Web Service Choreography and Orchestration for Business Process Management in conjunction with the Third International Conference on Business Process Management (BPM 2005), Nancy (France), September 2005*. Springer Verlag, 2005.

[13] B. D. Martino and O. F. Rana. Grid performance and resource management using mobile agents. In V. Getov, M. Gerndt, A. Hoisie, and A. M. B. Miller, editors, *Performance analysis and grid computing*, pages 251–263. Kluwer Academic Publishers, 2004.

[14] I. Müller, P. Braun, and R. Kowalczyk. A classification scheme for the integration of software agent and service-oriented paradigms. In L. Cavedon, R. Kowalczyk, Z. Maamar, D. Martin, and I. Müller, editors, *Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005) in conjunction with 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2005), Utrecht (The Netherlands), July 2005*, pages 57–60, 2005.

[15] A. Outtagarts, M. Kadoch, and S. Soulhi. Client-Server and Mobile Agent: Performances Comparative Study in the Management of MIBs. In A. Karmouch and R. Impey, editors, *Mobile Agents for Telecommunication Applications, Proceedings of the First International Workshop (MATA 1999), Ottawa (Canada), October 1999*, pages 69–81. World Scientific Pub., 1999.

[16] G. P. Picco. *Understanding, Evaluating, Formalizing, and Exploiting Code Mobility*. PhD thesis, Politecnico di Torino (Italy), 1998.

[17] A. Puliafito, S. Riccobene, and M. Scarpa. Which paradigm should I use? An analytical comparison of the client-server, remote evaluation and mobile agent paradigms. *Concurrency and Computation: Practice and Experience*, 13(1):71–94, 2001.

[18] G. Samaras, M. D. Dikaiakos, C. Spyrou, and A. Liverdos. Mobile Agent Platforms for Web-Databases: A Qualitative and Quantitative Assessment. In D. S. Milojicic, editor, *Proceedings of the First International Symposium on Agent Systems and Applications (ASA'99)/Third International Symposium on Mobile Agents (MA'99), Palm Springs (USA), October 1999*, pages 50–64. IEEE Computer Society Press, 1999.

[19] M. Straßer and M. Schwehm. A performance model for mobile agent systems. In H. R. Arabnia, editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), Las Vegas (USA)*, volume 2, pages 1132–1140. CSREA Press, 1997.

[20] W. Theilmann and K. Rothermel. Dynamic distance maps of the internet. In *Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Volume 1, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv (Israel), March 2000*, pages 275–284. IEEE Computer Society Press, 2000.

[21] J. E. White. Mobile agents. In J. Bradshaw, editor, *Software Agents*, pages 437–472. The MIT Press, Menlo Park, CA, 1996.