A Performance Model for Mobile Agent Systems

Markus Straßer
University of Stuttgart - IPVR
Breitwiesenstr. 20-22
70565 Stuttgart, Germany
Markus.Strasser@informatik.uni-stuttgart.de

Markus Schwehm
Tandem Computers Inc. - Loc 1
19333 Vallco Parkway
Cupertino, CA 96014
Schwehm_Markus@tandem.com

Abstract *A performance model for the interaction of agents in mobile agent systems is presented. Two interaction models, namely the remote procedure call and the agent migration are considered. Performance models for a single interaction are introduced, which are then used to derive a performance model for a sequence of interactions. This performance model can be used to evaluate the performance of any possible behaviour of an agent in a given scenario. The application of the performance model for a typical scenario in mobile computing shows that the optimal behaviour of an agent is achieved by a mixed sequence of remote procedure calls and agent migrations. The performance model is validated by measurements of interactions of real agents in the mobile agent system Mole.*

Keywords: mobile agents, performance model, remote procedure call, agent migration

1 Introduction

Mobile Agent Systems have received great attention in the last years as a new programming paradigm for widely distributed and heterogeneous systems. The basic concepts of agent systems are *locations* and *agents*. An agent system consists of a number of locations where computation can take place and where various services are provided. Agents are active entities which may move from location to location to meet other agents or to access services provided there. The mobility of the agents - i.e. their ability to migrate from one location to another - is the basic difference from other approaches for distributed systems.

It is often argued that the advantage of agent migration lies in the reduction of (expensive) global communication costs by moving the computation to the data ([5][6]). Although this argument is understandable from an intuitive point of view, not much work has yet been done to evaluate the performance of migration on a quantitative basis. A simple performance model was investigated in [3]. The evaluation of three scenarios was done in [4].

In this paper a performance model regarding network load and execution time is developed, which can help to identify situations for which agent migration is advantageous compared to remote procedure calls. This performance model is intended to help to decide which interaction model to be used in different scenarios of a mobile computation environment.

The paper is organized as follows: In Section 2 we describe two interaction models for mobile computation, namely remote procedure calls and agent migration. In Section 3, performance models for network load and execution time are derived for single interactions. In Section 4 the performance model is extended to include a sequence of interactions. In Section 5, the results are compared with those measured by our prototype implementation of a mobile agent system, Mole.

2 Interaction Models

Interaction between entities (objects, agents) in distributed systems which are located at different sites can take place in many different ways ([1]). In this paper we will confine on the remote procedure call as a global communication mechanism on one side and on agent migration via the network to the

communication partner followed by local communication on the other side.

2.1 Remote Procedure Calls (RPC)

Remote Procedure Calls are a widely used interaction mechanism in distributed systems. Basically a procedure is executed on a remote server, transferring the control flow (including some arguments) from the client to the server until the request is executed and the result is returned ([2]). Extensions to this synchronous concept include, among others, asynchronous RPC.

2.2 Agent Migration

Agent migration is a mechanism to continue the execution of an agent on another location ([5]). It includes the transport of agent code, execution state and data of the agent. In an agent system, the migration is initiated on behalf of the agent and not by the system (e.g. for load balancing purposes). The basic motivation for migration is to move the computation to a data server or a communication partner in order to reduce network load by accessing a data server or communication partner by local communication.

2.3 Remote Execution

Remote execution in the context of mobile agents is a mechanism to start (rather than to continue) the execution of an agent on another location. It includes only the transport of agent code and some parameters. Due to the similarity of remote execution and agent migration regarding their communication needs, remote execution is not further considered in this paper.

3 A Single Interaction

In this section a single client/server-style interaction is considered. The following simplifying assumptions are made: The interaction partners and the amount of communication for request and reply in each interaction is known in advance. Average values for delay and throughput are considered. The time for marshalling and unmarshalling (i.e. transformation of entities in a transport format and back) increases linear with the size of the data to be sent. All locations execute jobs with the same speed.

3.1 Interaction by RPC

In the context of agent interaction, the RPC is used to call procedures (methods) that are provided by the communication partner (e.g. a service agent). A (classical) RPC includes binding to the server, marshalling, transfer, unmarshalling of the request parameters, execution of the request, and marshalling, transfer and unmarshalling of the reply. With the above mentioned simplifying assumptions, the time for binding can be omitted since communication partners are known in advance and the time for execution of the request can be omitted since not influenced by the interaction model. Marshalling is dependent of the size of the request parameters only. The performance model therefore concentrates on the communication dependent part of the RPC.

The network load B_{RPC} (in bytes) for a simple remote procedure call from location L_1 to location L_2 consists of the size of the request B_{req} and the size of the reply B_{rep}:

$$B_{RPC}(L_1, L_2, B_{req}, B_{rep}) = \begin{cases} 0 & \text{if } L_1 = L_2 \\ B_{req} + B_{rep} & \text{else} \end{cases}$$

The execution time T_{RPC} for a simple remote procedure call from location L_1 to location L_2 consists of the time for marshalling and unmarshalling of request and reply (factor μ) plus the time for the transfer of the data on a network with throughput $\tau(L_1,L_2)$ and delay $\delta(L_1,L_2)$

$$T_{RPC}(L_1, L_2, B_{req}, B_{rep}) = 2\delta(L_1, L_2) + \left(\frac{1}{\tau(L_1, L_2)} + 2\mu\right) B_{RPC}(L_1, L_2, B_{req}, B_{rep})$$

3.2 Interaction by Agent Migration

In this section, an interaction is performed by the migration of the agent to the communication partner, a local or remote procedure call, the processing of the data received and the transfer of the processed data back to the source location. A (classical) migration includes marshalling, transport and unmarshalling of code, data and execution state of the agent to the server. With the same simplifying assumptions as above, the time for the execution of the procedure call can be omitted. Mar-

shalling increases linear with the size of data and execution status of the agent, while the code of the agent is already stored in transport format and is only transferred on demand (i.e. if the code is not yet available at the server).The agent consists of B_{code} bytes of code, B_{data} bytes of data and B_{state} bytes of execution state and is described by the triple B_A=(B_{code}, B_{data}, B_{state}). The size of the request to the procedure B_{req} is yet contained in B_{data}. The size of the reply from the procedure B_{rep} is reduced by remote processing to $(1\text{-}\sigma)B_{rep}$ with $(0 \le \sigma \le 1)$ by the agent where σ models the selectivity of the agent.

The network load for the migration of an agent A from location L_1 to location L_2 is calculated by

$$B_{Mig}(L_1, L_2, B_A) = \begin{cases} 0 & \text{if } L_1 = L_2 \\ P(B_{cr} + B_{code}) + B_{data} + B_{state} & \text{else} \end{cases}$$

where P denotes the probability that the code is not yet available at location L_2 and B_{cr} is the size of the request from L_2 to L_1 to transfer the code. If the agent additionally sends back a reply message to location L_1, the network load amounts

$$B_{MR}(L_1, L_2, B_A, \sigma, B_{rep}) = B_{Mig}(L_1, L_2, B_A) + \begin{cases} 0 & \text{if } L_1 = L_2 \\ (1-\sigma)B_{rep} & \text{else} \end{cases}$$

where B_{rep} is the size of the reply and σ denotes the selectivity of the agent.

The corresponding execution time including delay δ, throughput τ and marshalling overhead μ for a single agent migration from location L_1 to location L_2 is described by

$$T_{Mig}(L_1, L_2, B_A) = (1 + 2P)\delta(L_1, L_2) + \frac{B_{Mig}(L_1, L_2, B_A)}{\tau(L_1, L_2)} + \begin{cases} 0 & \text{if } L_1 = L_2 \\ 2\mu(B_{data} + B_{state}) & \text{else} \end{cases}$$

and including the reply message back to location L_1, the execution time amounts

$$T_{MR}(L_1, L_2, B_A, \sigma, B_{rep}) = T_{Mig}(L_1, L_2, B_A) + \delta(L_1, L_2) + \begin{cases} 0 & \text{if } L_1 = L_2 \\ \left(\frac{1}{\tau(L_1, L_2)} + 2\mu\right)(1-\sigma)B_{rep} & \text{else} \end{cases}$$

3.3 Evaluation of a Single Interaction

To evaluate a single remote procedure call and a single agent migration based on these simple models, we consider the following scenario: The agent consists of B_{code}=39kBytes of code, B_{data}= 5kBytes of data and B_{state}=5kBytes of execution state. With a probability of P=10% the code of the agent is not yet available at the remote location, in this case the transmission of the code has to be initiated by a code request message of size B_{cr}=1kByte. The request size of the interaction is B_{req}=1kByte. Figure 1 compares the network load of the remote procedure call (B_{RPC}) to agent migration with reply (B_{MR}) for a fixed reply size of B_{rep}=25kBytes while varying the selectivity σ between 0% and 100%.

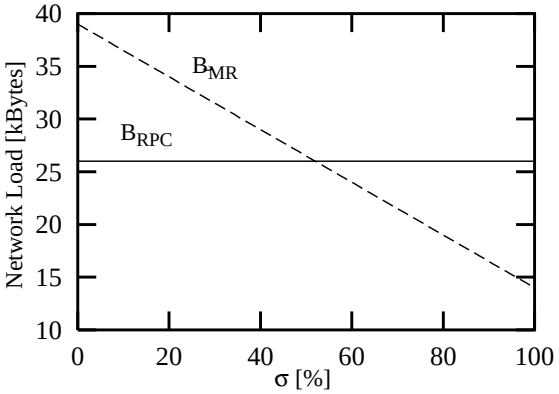

Figure 1: Network load versus selectivity

Figure 2 compares the network load of remote procedure call (B_{RPC}) and agent migration with reply (B_{MR}) for a fixed selectivity of σ=0.9 while the reply size B_{rep} is varied between 0 and 30kBytes.

The diagrams in Figures 1 and 2 show that the usage of agent migration rather than the usage of remote procedure calls reduces network load only if the size of the reply is large and/or the agent has a large selectivity.

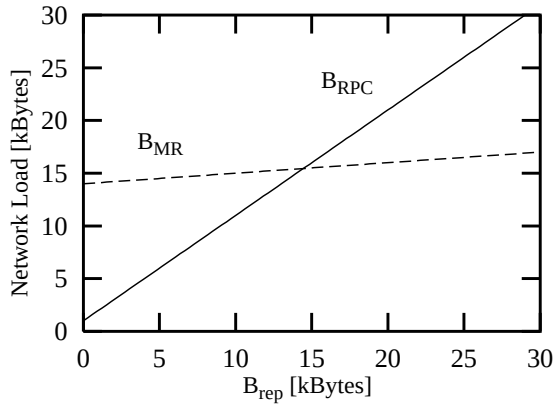

Figure 2: Network load versus reply size

Figures 3 and 4 show the corresponding diagrams for the execution time with assumed network characteristics delay δ=30ms, throughput τ=400kBytes/s and no marshalling overhead (μ=0s/kByte).

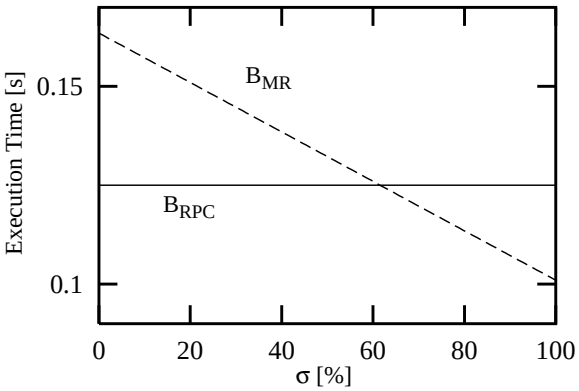

Figure 3: Execution time versus selectivity

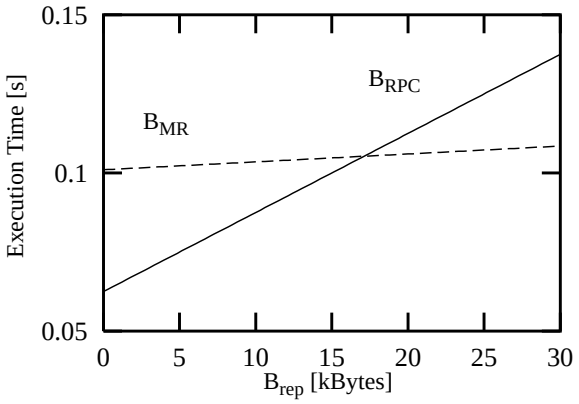

Figure 4: Execution time versus reply size

While the overall behaviour of these graphs imply the same conclusions, it should be noted that the break-even point between remote procedure call and agent migration for network load differs from the break-even point for execution time. For example, the break-even point for network load in Figure 1 is reached at σ=52%, while the break-even point for execution time in Figure 3 is reached at σ=62%.

4 A Sequence of Interactions

In this section a sequence of several interactions with different interaction partners on different locations is considered. The following simplifying assumptions are made: The sequence of interaction partners and their locations, as well as each request size, reply size, selectivity and number of communications per location is known in advance. Furthermore it is assumed that average delays and average throughput in a possibly inhomogeneous network are known for every possible interconnection.

Let $S=(I_1,...,I_n)$ be a sequence of interactions. The *i*-th interaction is described by

$$I_i = \{R_i, m_i, B_{req_i}, B_{rep_i}, \sigma_i\}$$

where R_i is the remote location with which the communication should take place. Each communication consists of m_i (local or remote) procedure calls with request size B_{req_i}, reply size B_{rep_i} and selectivity σ_i. The size of the agent after interaction *i* is modelled for i=0,...,n by

$$B_{A_i} = (B_{code_i}, B_{data_i}, B_{state_i})$$

where B_{code_i} and B_{state_i} remain fixed while

$$B_{data_i} = B_{data_{i-1}} + m_i(1-\sigma_i)B_{rep_i}$$

The mobility behaviour of the agent is described by a destination vector $D=(D_0,...,D_n)$. For the *i*-th interaction the agent moves to destination location D_i. Thus migration takes place between *(i*-1*)*-th and *i*-th interaction only if $D_i \neq D_{i-1}$. Please note that the agent need not migrate to location R_i where the next interaction takes place.

The network load and execution time for a mixed sequence of remote procedure calls and agent migrations are calculated as follows:

$$B_{Seq}(S, D, B_{A_0}) =$$

$$\sum_{i=1}^{n} (B_{Mig}(D_{i-1}, D_i, B_{A_{i-1}}) + m_i B_{RPC}(D_i, R_i, B_{req_i}, B_{rep_i}))$$

$$T_{Seq}(S, D, B_{A_0}) =$$

$$\sum_{i=1}^{n} (T_{Mig}(D_{i-q}, D_i, B_{A_{i-1}}) + m_i T_{RPC}(D_i, R_i, B_{req_i}, B_{rep_i}))$$

4.1 Evaluation of Scenario 1

To evaluate a sequence of interactions based on this performance model, we considered a typical situation in mobile computing, namely the usage of a laptop or personal digital assistant (PDA) at location L_0 that only has wireless low bandwidth access to the Internet. The characteristics of this rather inhomogeneous network are assumed as follows: Each interaction between locations x and y inside the Internet has a delay of $\delta(x,y)$=10ms and a throughput of $\tau(x,y)$= 400kBytes/s. Each interaction between the PDA and the Internet has a delay of $\delta(x,y)$=120ms and a throughput of $\tau(x,y)$=50kBytes/s.
The mobile agent starts on the PDA at location L_0 and has to interact with four locations L_1...L_4 before returning the result to L_0 (as illustrated by thin arrows in Figure 5).

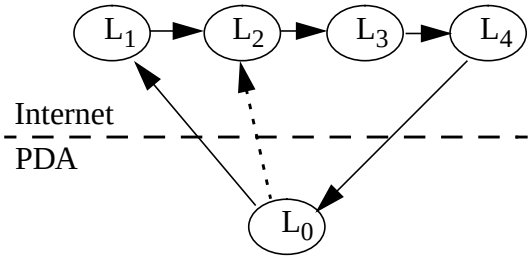

Figure 5: Mobile computing scenario

The amount of communication with locations L_2 and L_4 is much larger than with the other locations. In particular, in scenario 1 a mobile agent with B_{code}=10kBytes, B_{data}=5kBytes, B_{state}=5kBytes is considered. It is assumed that the code of the agent is not available at the remote locations (P=100%) and that the size of the agent's data does not increase (σ=1). The mission of the agent is to process the sequence of interactions S_1 listed in Table 1.

Table 1: Sequence on interactions S_1

i	R_i	m_i	B_{req_i}	B_{rep_i}	σ_i
1	L_1	1	50	2000	1
2	L_2	1	500	4000	1
3	L_3	1	50	2000	1
4	L_4	1	500	4000	1
5	L_0	1	500	10	1

The performance model for interaction sequences was now used to evaluate the execution time for all possible agents regarding their mobility behaviour. The diagram in Figure 6 shows the execution time for all possible destination vectors D sorted on the horizontal axis according to the number of migrations involved.

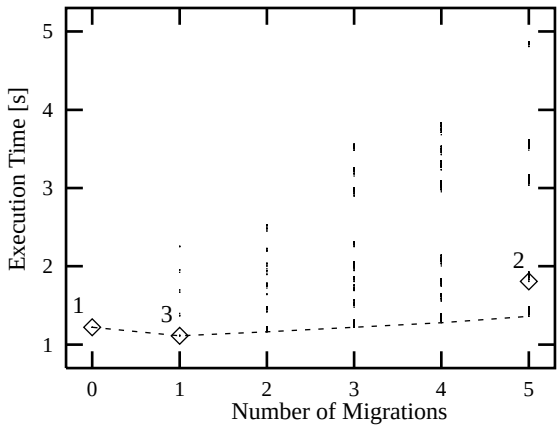

Figure 6: Execution times for scenario 1

Three of the execution times are marked in the diagram and the corresponding values of the destination vector D, the execution time T_{Seq} and the network load B_{Seq} are listed in Table 2 for further inspection. Evaluation 1 indicates the execution time of an agent that only uses RPC and no migration at all. Evaluation 2 on the other hand uses migration to the location of the next interaction partner for each of the five interactions. The execution

Table 2: Performance values for scenario 1

#	Destin. Vector D	T_{Seq}	B_{Seq}
1	L_0,L_0,L_0,L_0,L_0,L_0	1.2220	13100
2	L_0,L_1,L_2,L_3,L_4,L_0	1.8075	105000
3	L_0,L_2,L_2,L_2,L_2,L_2	1.1117	30110

time is much larger than for the RPC-only evaluation 1 due to the much larger network load involved. Figure 6 furthermore shows that evaluation 2 is not the fastest solution with five migrations. Due to the low throughput of the wireless link to the PDA, it is still better to make a (useless) fifth migration to any other internet location (L_1 - L_4) before interaction I_5 and then to use an RPC to transmit the results back to location L_0. Interestingly, the shortest execution time is achieved with evaluation 3 by an agent that uses migration exactly once. The destination vector for evaluation 3 shows that the corresponding agent migrates to location L_2 before interaction I_1 (i.e. not to the interaction partner of I_1 but to the first location with a large amount of communication) as indicated by the dotted arrow in Figure 5 and remains there until to the end of the sequence. Again the network load is larger than for evaluation number 1, but the intelligent usage of a single migration reduces execution time compared to the RPC-only evaluation 1.

4.2 Evaluation of Scenario 2

In the second scenario the same values for the agent (B_A, P and σ) are used, but the sequence of interactions changes slightly according to Table 3. In particular only interactions 2 and 4 are modified by replacing one communication and large request/reply size with ten communications and small request/reply size, so that the total amount of data transferred is not changed.

Table 3: Sequence of interactions S_2

i	R_i	m_i	B_{req_i}	B_{rep_i}	σ_i
1	L_1	1	50	2000	1
2	L_2	10	50	400	1
3	L_3	1	50	2000	1
4	L_4	10	50	400	1
5	L_0	1	500	10	1

The diagram in Figure 7 shows the evaluation of the execution time for all possible destination vectors, again sorted according to the number of migrations involved. The smallest execution time (evaluation number 4) is achieved by an agent that uses migration twice. Table 4 lists the corresponding values of the destination vector, execution time and network load.

Table 4: Performance values for scenario 2

#	Destin. Vector D	T_{Seq}	B_{Seq}
1	L_0,L_0,L_0,L_0,L_0,L_0	5.5420	13100
2	L_0,L_1,L_2,L_3,L_4,L_0	1.8075	105000
3	L_0,L_2,L_2,L_2,L_2,L_2	1.2917	30110
4	L_0,L_2,L_2,L_2,L_4,L_4	1.1629	46610

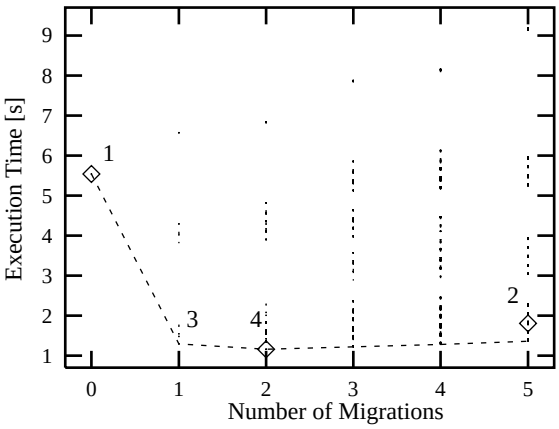

Figure 7: Execution times for scenario 2

Evaluation 1 shows the values for the agent using RPC only. Although the amount of data transmitted is the same as in scenario 1, the execution time increased heavily due to the larger number of communications in scenario 2 because each additional RPC increases the execution time by two delays. Evaluation 2 for the agent using migration to the next interaction partner for each interaction does not change compared to scenario 1 because all communications are realized by local procedure calls. Nevertheless in scenario 2 the execution time of the always migrating agent has become better

than the execution time for the RPC-only agent. The execution time for the agent in evaluation 3 with one single migration to location L_2 is in scenario 2 not optimal any more. The additional execution time for the increased number of remote procedure calls to location L_4 supersedes the amount of time needed for a second migration to location L_4 before interaction I_4 in order to realize these communications by local procedure calls, as it is done in evaluation 4.

5 Experimental Validation

To validate the introduced performance model, we have performed measurements of the execution time of mobile agents running on Mole, an implementation of a mobile agent system.

5.1 The Mobile Agent System Mole

Mole is a mobile agent system implemented in Java ([7][8]). It provides (Java-)agents the ability to migrate to other locations and to communicate with other mobile agents or to static service agents via local or remote procedure calls or by messaging. In Mole, only a "restricted migration" is implemented, which transfers only the code and the data of the agent, but not the execution state of the agent to the destination location. The code is transported to the destination location only if not yet available there.

5.2 Experimental Setup

Since we did not have access to a mobile device with a running mobile agent system, the mobile computing scenario from Section 4 was imitated by placing location L_0 on a host in the US (ICSI, Berkeley) and a cluster of four locations L_1 to L_4 on hosts on a local area network in Germany (IPVR, University of Stuttgart). This setup provides a slow and low-bandwidth connection from location L_0 to the cluster compared to the fast and high-bandwidth connection inside the cluster. The values for delay $\delta(x,y)$, throughput $\tau(x,y)$ and the marshalling overhead μ for previous interactions are measured by the mobile agent system and are accessible by the agents for further usage.

Two interaction sequences were defined with similar characteristics as the interaction sequences in Section 4. Table 5 lists interaction sequence S_3

Table 5: Sequence of interactions S_3

i	R_i	m_i	B_{req_i}	B_{rep_i}	σ_i
1	L_1	1	700	3000	1
2	L_2	1	3500	15000	1
3	L_3	1	700	3000	1
4	L_4	1	3500	15000	1
5	L_0	1	3000	700	1

which is characterized by a larger amount of communication with service agents on location L_2 and L_4 than with those on locations L_1 and L_3. The resulting scenario corresponds to scenario 1 of Section 4.1. Table 6 lists interaction sequence S_4

Table 6: Sequence of interactions S_4

i	R_i	m_i	B_{req_i}	B_{rep_i}	σ_i
1	L_1	1	700	3000	1
2	L_2	5	700	3000	1
3	L_3	1	700	3000	1
4	L_4	5	700	3000	1
5	L_0	1	3000	700	1

where the single large communication with agents on location L_2 and L_4 is replaced by five smaller communications such that the total amount of bytes transmitted is not changed. This scenario corresponds to scenario 2 of Section 4.2.

To undertake the measurements, a static agent is started on location L_0. The static agent starts other mobile agents which have to interact with service agents on locations L_1 to L_4 according to the mobile computing scenario in Figure 5. The static agent measures the time from the initialization of a mobile agent until to the delivery of the corresponding reply message by RPC. The characteristics of the mobile agents are B_{code}=10kBytes, B_{data}=32kBytes, B_{state}=0Bytes (due to the restricted migration used in the Mole system). The agent code does never need to be transmitted (P=0%)

and the size of the agent data does not change (σ=1).

To carry out their mission, the mobile agents followed one of three mobility strategies: The 'RPC-only'-agent remained on location L_0 and used remote procedure calls for each interaction. The 'always-migrate'-agent always migrates to the next interaction partner and then uses local procedure calls. The 'optimized'-agent uses the performance model for the execution time T_{Seq} to decide when and to which location it should migrate.

5.3 Experimental Results

The execution times for interaction sequence S_3 are shown in Table 7. The measurements are averaged over 50 runs of the mobile agent for each of the three mobility strategies. Similar to the results obtained for S_1, the 'RPC-only' strategy is faster than the 'always-migrate' strategy and the 'optimized' strategy offers only a small improvement. Using the 'optimized' strategy the mobile agent migrated exactly once in 49 of the 50 runs. 47 times it migrated to location L_2 and 2 times to location L_4, an observation which reflects the dynamically changing network load measured by the mobile agent system and used by the 'optimized' strategy.

Table 7: Measurements f. interaction sequence S_3

mobility strategy	average time [ms]	standard deviation [ms]
'RPC-only'	7501	748
'always-migrate'	9793	1140
'optimized'	7462	1341

The execution times for the three mobility strategies applied to interaction sequence S_4 and averaged over 50 runs of the mobile agent are listed in Table 8. As expected, here the 'always-migrate' strategy performs better than the 'RPC-only' strategy and the 'optimized' strategy offers another improvement. Using the 'optimized' strategy, the mobile agent migrated exactly once in 30 cases (11 times to L_2, 6 times to L_3 and 13 times to L_4) and in 20 cases the mobile agent migrated exactly two times (14 times to L_2 and L_4; 6 times to L_3 and L_4). Again, the dynamically changing measured values for delay and throughput explain this observation.

Table 8: Measurements f. interaction sequence S_4

mobility strategy	average time [ms]	standard deviation [ms]
'RPC-only'	19127	1516
'always-migrate'	11394	1414
'optimized'	10953	1341

6 Conclusion

We have introduced a performance model for mobile agent systems where agents can alternatively use remote procedure calls or agent migration for the interaction with partners on different locations. The model was first used to identify situations where a single agent migration has advantages compared to a single remote procedure call. This is basically the case when the amount of data to be processed is large compared to the size of the agent and if the selectivity of the agent, i.e. the ability of the agent to reduce the size of the reply by remote processing, is high. Then the model was extended to describe a sequence of interactions. From this model the conclusion can be drawn that an alternating sequence of remote procedure calls and agent migrations performs better than a pure sequence of remote procedure calls or a sequence of agent migrations. This result was confirmed by measurements on a prototype implementation of a mobile agent system, Mole. The performance model for the execution time was also used to optimize the mobility behaviour of the mobile agent in a given interaction scenario. The optimization was successful although dynamic fluctuations of measured model parameters like network delay and throughput have weaken this result. The performance model could thus be a building block for the optimization of the mobility behaviour of mobile agents.

7 References

[1] J. Baumann, F. Hohl, N. Radouniklis, K. Rothermel and M. Straßer. Communication Concepts for Mobile Agent Systems. In: *Mobile Agents, Proc. 1st Int. Workshop, MA '97*. Springer, 1997.

[2] A.D. Birrell and B.J. Nelson. Implementing Remote Procedure Calls, In *ACM Trans Computer Systems*, Vol. 2, pp. 39-59. 1984

[3] A. Carzaniga, G.P. Picco and G. Vigna. Designing Distributed Applications with Mobile Code Paradigms. To appear in: *Proc. 19th Int. Conf. on Software Engineering*, Boston. 1997

[4] T.-H. Chia and S. Kannapan. Strategically Mobile Agents. In: *Mobile Agents, Proc. 1st Int. Workshop, MA '97*. Springer, 1997.

[5] General Magic, Inc. The Telescript Language Reference. http://www.genmagic.com-Telescript/TDE/TDEDOCS_HTML/telescript.html, 1996

[6] C.G. Harrison, D.M. Chess and A. Kershenbaum. Mobile Agents: Are they a good idea? IBM Research Report, 1995

[7] Project Mole, http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole.html

[8] M. Straßer, J. Baumann and F. Hohl. Mole - A Java Based Mobile Agent System. In: J. Baumann *et al* (Eds.) Proc. 2nd ECOOP Workshop on Mobile Object Systems, dpunkt-Verlag 1996