# A Selection Algorithm for an Efficient Interaction Pattern out of Paradigms

Hyeok Chan Kwon
Dept. of Computer Science
ChungNam National University
Kung-Dong, Taejon 305-764, Korea
+82-42-821-5442
hckwon@cs.cnu.ac.kr

Sooho Sohn
ETRI
Taejon, Korea
+82-42-860-6441
shsohn@etri.re.kr

Woo Jong Yoo
Dept. of Computer information processing
Taejon health sciences college
Taejon, Korea
+82-42-630-5953
wjyoo@tjhealth.ac.kr

Sung In Lee
Dept. of Office Automation
Taejon health sciences college
Taejon, Korea
+82-42-630-5942
silee@tjhealth.ac.kr

Tae Gun Kang
Dept. of computer Science
ChungNam National University
Kung-Dong, Taejon 305-764, Korea
+82-42-821-5442
tgkang@nspplab.cnu.ac.kr

Kwan Jong Yoo
Dept. of computer Science
ChungNam National University
Kung-Dong, Taejon 305-764, Korea
+82-42-821-5442
kjyoo@cs.cnu.ac.kr

## ABSTRACT

A major benefit provided by mobile agent is the capability to reduce network traffic by moving agent itself to server node. But the question of whether the system using mobile agent is bringing significant benefits to the performance of distributed applications against traditional approach is an open one. Various parameters must be considered to evaluate performance of a paradigm used in developing distributed application. Performance of distributed application is affected not only by a paradigm but also by the migration pattern of the paradigm. In this paper, we present performance evaluation model for mobile agent against RPC(Remote Procedure Call) that is a traditional client-server architecture. And on the basis of this model, we present efficient selection algorithm for an interaction pattern from paradigms. We also present simulation results of this algorithm.

## Keywords

mobile agent, RPC(Remote Procedure Call), locker pattern, interaction pattern

## 1. INTRODUCTION

A mobile agent is an executing program that can migrate during execution from machine to machine. In RPC(Remote Procedure Call) and multi agent, the tasks are performed by global communication with remote site, whereas mobile agent system performs the task by migrating a whole computational component, together with its state, the code, and some resources[1][2].

Many researchers suggest that a major benefit provided by mobile code is the capability to reduce network communication by moving client's knowledge close to server's resources, thus accessing them locally. By moving to the location of information resource, the agent can search the resource locally, eliminating the transfer of intermediate results across the network and reducing end-to-end delay. Recently, for this benefit of mobile agent, the demand of applying mobile agent to distributed systems such as information retrieval, network management, and electronic commerce has been increased.

But the question of whether the system using mobile agent is bringing significant benefits to the performance of distributed applications against traditional approach is an open one. Various parameters must be considered to evaluate performance of a paradigm used in developing distributed application. Performance of distributed application is affected not only by a paradigm but also by the migration pattern of the paradigm. So we need quantitative performance model to decide an appropriate interaction model in developing distributed application. In this paper, we present performance evaluation model for mobile agent and RPC that is, a traditional client-server architecture. And on the basis of this model, we present an efficient selection algorithm for an interaction pattern from paradigms. We also present simulation results of this algorithm. This paper is intended to help to decide which interaction model to be used in different distributed applications.

There is no interaction pattern that is better than others in absolute terms. The choice of the interaction pattern must be performed on a case-by-case basis, according to the type of application. For performance evaluation, parameters such as CPU costs, memory usage and network traffic etc. should be considered. In this paper, we only concern network traffic. And we choose a simple data mining for target application.

For this model, we consider three kinds of interaction patterns: RPC, mobile agent and another mobile agent applied locker pattern.

In Locker pattern[5], mobile agent temporarily stores data in private. In this way it can avoid bringing data that at the moment are not needed. On a later occasion, the stored data can be sent to client node, or another mobile agent migrates nodes to gather the data stored in private area. We assumed the former case. The three paradigms considered in this paper are shown in Fig. 1.

The paper is organized as follows. Section 2 presents performance evaluation model for data mining application. Section 3 presents selection algorithms. Our simulation results are presented in section 4. Finally conclusion is given in section 5.

## 2. PERFORMANCE EVALUATION MODEL

For Simplicity we make the following assumptions:

- The locations of client and server node are different.

- Uniform network
- There exist N network nodes.
- All requests have a fixed length.
- The average document size is different at each node.
- The probability r that a document contains useful information is fixed. ($0 \leq r \leq 1$)
- In case of RPC, client requests the document header to each server. In this time, the client can find whether the document contains useful information or not by document header. If the document is considered to contain useful information, the client requests it to the server.

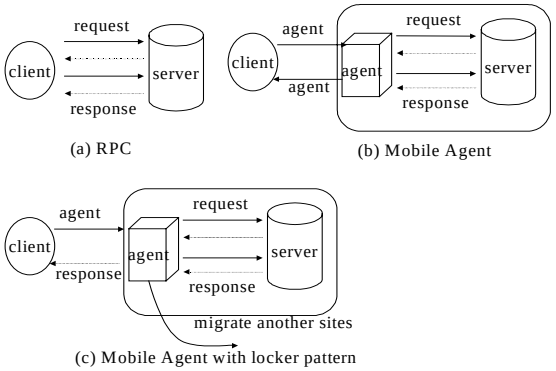


Figure 1. Three paradigms

| parm. | | unit |
|---|---|---|
| N | the number of network node | number |
| $M_j$ | the number of documents of node j | number |
| r | the probability that a document may contain useful information | $0 \leq r \leq 1$ |
| req | size of request message | bytes |
| $rep_{hj}$ | size of reply message (header) of node j | bytes |
| $rep_{dj}$ | average size of document | bytes |
| $\delta$ | network delay | ms |
| $\beta$ | average network bandwidth | KB/sec |
| Ma | size of mobile agent | KB |

Table 1. parameters

Parameters are shown in table 1. The size of mobile agent to transfer is shown in equation 1.

$$Ma = DMA + (DMA/payload)*(IPheader + TCPheader) \quad (1)$$

$$DMA = ATP\_header + SMA$$

$$SMA = M\_state + M\_code + M\_data$$

where SMA denotes the size of mobile agent itself that contains agent code, data and state. For transferring agent, agent is attached at the next of an ATP(Agent Transfer Protocol) header. And ATP message is segmented into TCP messages. The payload in equation 1 is TCP payload. Each TCP message is attached at the next of a TCP header and encapsulated into an IP packet[3].

## 2.1 Eliminating duplicate data

In this section, we present a couple equations that calculate duplicate data at each node. The additional parameters are shown in Table 2.

TD is the total number of documents of all nodes. VD indicates the number of documents that were found in the previous. UR is the ratio of unique data out of all data. In this paper, we assume there is no duplicate data within a node. The number of unique data is represented as TD*UR, and NUR is obtained by subtracting one from UR. The equation 2 shows the average number of duplicate data at node j.

| Parameter | | unit |
|---|---|---|
| UR | Unique Rate | $0 \leq UR \leq 1$ |
| NUR | Non Unique Rate | $0 \leq NUR \leq 1$ |
| TD | Total Document | number |
| VD | Visited Document | number |

Table 2. Additional parameter

$$NMj = 0 \quad if \quad j = 1 \qquad (2)$$

$$M_j \times \{VD/(TD - Mj) \times (2 \times NUR)\} \quad if \quad j \rangle 1$$

Table 3 shows the process of eliminating duplicate data when network node is 5 (N=5), and unique rate is 0.8 (UR=80%). At each node, the numbers of documents are 10, 30, 20, 10 and 30 respectively.

With header information, we could distinguish duplicate data in some degree by comparing one document's header to another, not by their contents. But, in case of mobile agent, much more data can be eliminated if we compare whole content of one document to the other, because mobile agent has accumulated documents while migration.

The method of eliminating duplicate data at each paradigm is as follows: In RPC, a client can discriminate whether a data is duplicate one or not by header information that the client has got from servers. First, the client receives header list from server, and then eliminates unnecessary data and duplicate data, and then requests the rest documents to server. Mobile agent easily discriminate duplicate data by moving agent itself with collected document. In locker pattern, to discriminate duplicate data, mobile agent accumulates headers into data area of itself, and then it migrates to another site.

| Visit node | $M_j$ | # of doc. that removed (approx.) | | | # of selected doc. (approx.) | |
|---|---|---|---|---|---|---|
| | | | Equation | % | | |
| 1 | 10 | 0 | 0 | 0 | 10 | M |
| 2 | 30 | 1.7 | 30*{10/(100-30) * (2*0.2)} | 5.7 | 28.3 | M-1.7 |
| 3 | 20 | 4 | 20*{40/(100-20) * (2*0.2)} | 20 | 16 | M-4 |
| 4 | 10 | 2.6 | 10*{60/(100-10) * (2*0.2)} | 26.7 | 7.4 | M-2.6 |
| 5 | 30 | 12 | 30*{70/(100-30) * (2*0.2)} | 40 | 18 | M-12 |
| Tot. | 100 | 20.3 | | | 79.7 | |
| j | $M_j$ | | $M_j$*{DB/(TD-$M_j$)*(2* NUR)} | | | |

Table 3. The process of eliminating duplicate data

## 2.2 Model for RPC(Remote Procedure Call)

Equation 3 and 4 shows network load and network execution time at a node j respectively. $UM_j$ is the number of documents after eliminating duplicate data, that is, the number obtained by subtracting $NM_j$ from $M_j$ from the equation 2.

$$L_{RPC-S} = (1 + rUM_j)req + UM_j(rep_h + rrep_{dj}) \qquad (3)$$

$$T_{RPC-S} = \frac{L_{RPC-S}}{\beta} + (2 + 2UM_j r)\delta \quad (4)$$

Client requests headers just once at each node while it requests as many documents as it needs. For each node, it requires delay of two to request and reply document header, and delay of $2rUM_j$ for document bodies. Client does not have to receive all documents from server because it can decide whether the document is necessary or not by looking up its header respectively.

## 2.3 Model for Mobile agent

Equation 5 and 6 shows network load and network execution time for migrating a mobile agent to node j respectively.

$$L_{MA} = 2Ma + \sum_{k=1}^{t} (rUM_k rep_{dk}) + rrep_{dj} UM_j \quad (5)$$

(t is number of node that is visited before by mobile agent)

$$T_{MA} = 2\delta + \frac{L_{MA}}{\beta} \quad (6)$$

Two times of network delay are required for agent migration. In the equation 5, $\sum_{k=1}^{t} (rUM_k rep_{dk})$ stands for the size of previously accumulated data, and $rrep_{dj} UM_j$ is the amount of accumulated data from the current node.

## 2.4 Model for locker pattern

Equation 7 shows the network load, and equation 8 for the network execution time when mobile agent applied locker pattern migrates to node j.

$$L_{MA(L)} = MaH + rUM_j rep_{dj} \quad (7)$$

$$MaH = Ma + \sum_{k=1}^{t} (rUM_k rep_{hk})$$

(t is number of node that is visited before by locker pattern)

$$T_{MA(L)} = \frac{L_{MA(L)}}{\beta} + (1 + UM_j r)\delta \quad (8)$$

As for discriminating duplicate data, locker pattern accumulates each document's header into the data area of mobile agent. The size of accumulated headers is relatively small compared to the total size of data to be moved. In locker pattern, the amount of $rUM_j rep_{dj}$ network load is required for transferring collected document to server.

The model devised so far is meaningful only for a uniform network. For non-uniform network environment, it requires somewhat different model against communication costs. We are going to design a model for non-uniform networks in the future.

# 3. A SELECTION ALGORITHM

The performance of each paradigm hinges on its parameter value. The major parameters are network bandwidth, size of accumulated data, number of interaction with server, size of mobile agent, and so on. In mobile agent, as we can see from the equation 5 and 6, it consumes a little time to make the network connection, whereas it consumes much more time to transfer mobile agent and its accumulated data. But in RPC as we can see the equation 3 and 4, it normally needs to make many global communications, which produces a great deal of delay, whereas it needs far less time in migrating over network. Mobile agent, therefore, shows better performance than RPC does when it requires frequent global communications or when the condition of given network is good as well. On the contrary, RPC shows performance than mobile agent when it requires fewer global communications or the condition of given network is not good. The number of global communications depends on the number of documents at each node in case of data mining applications. The performance of both mobile agent and locker pattern is affected by the size of mobile agent.

## 3.1 The criterion of selective migration

We simplify notations given in section 2 for presenting better insight into the meaning of the formulae.

Mobile agent performs better than RPC if $T_{RPC-S} > T_{MA.}$ After substituting and elaborating equations 3-5 in $T_{RPC-S} > T_{MA}$ in criterion of network delay times, we obtain

$$rUM_j\delta \rangle \frac{2Ma + AD(t) - (rreq + rep_h)UM_j - req}{\beta} \quad (9)$$

AD(t) means $\sum_{k=1}^{t} (rUM_k rep_{dk})$.

RPC performs better than locker pattern if $T_{RPC-S} > T_{MA(L)}$. After substituting and elaborating equations 3,4,7,8 in $T_{RPC-S} > T_{MA(L)}$ in criterion of network delay times, we obtain

$$(1 + rUM_j)\delta \rangle \frac{MaH - (rreq + rep_h)UM_j - req}{\beta} \quad (10)$$

Mobile agent performs better than locker pattern if $T_{MA} > T_{MA(L)}$. After substituting and elaborating equations 5-8 in $T_{MA} > T_{MA(L)}$, we obtain

$$\frac{AD(t) - Ma}{\beta} \rangle (rUM_j\delta - 1)\delta \quad (11)$$

## 3.2 Algorithms

An algorithm selecting an efficient interaction pattern from the given paradigms is presented in this section. In the following algorithm, migration function is inferred from formulae in section 2 and 3.1.

■ Case I : fixed node sequence to visit

input : node list, *other parameters* ....
output : selected_paradigm_list, agent_position_list

```
set current_agent_position as client_node
for node# from 1 to last_node
  choose paradigm which has minimum execution time by
                                        migration function
      Add selected_paradigm into selected_ paradigm_ list
      If selected_paradigm = locker pattern
          update accmulated_header value
          set current_agent_position as node#;
      If selected_paradigm = MA
          update AD(t) value
      set current_agent_position as node#;
```

End loop;

■ Case II : not fixed node sequence to visit

In case of non-fixed node sequence to visit, the better performance is achieved by changing node sequence to visit. The algorithms are as follows.

input : node list, *other parameters ....*
output : table that composed of pair of (selected_node, paradigm)

```
set current_agent_position as client_node
While unvisited_node = none
  visit node by increasing order of total document size
  for each node
    choose paradigm which has minimum execution time by
                                                migration function
      Add selected_paradigm into selected_paradigm_list
      If selected_paradigm = locker  pattern
           update accmulated_header value
           set current_agent_position as node#;
      If selected_paradigm = MA
           update AD(t) value
           set current_agent_position as node#;
 end loop;
```

End loop;

## 4. SIMULATION

In this section, we present simulation results of each algorithm. We used Visual C++ 6.0 for simulation, and considered following parameter values : number of node N=8, network delay $\delta$ =20ms, network bandwidth $\beta$ =300kbytes/s, r=40%, UR=80% and size of mobile agent was 15KB. The size of request and reply messages of header were 50, 60 bytes respectively. At each node, the average number of documents and their sizes are shown in table 4.

| node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| doc. size(KB) | 2 | 5 | 10 | 3 | 5 | 15 | 1.5 | 3 |
| # of doc. | 6 | 7 | 2 | 30 | 50 | 5 | 20 | 40 |

Table 4. Parameter values

■ Case I : fixed  node sequence to visit

The migration sequence after applying our algorithm with the above assumptions is shown in table 5. The table 6 shows migration sequence when applying algorithm in reverse for comparison purpose.

| visit node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | C |
|---|---|---|---|---|---|---|---|---|---|
| Paradigm | L | L | R | M | M | L | L | L | M |
| loc. Of agent | 1 | 2 | 2 | 4 | 5 | 6 | 7 | 8 | C |

Table 5. After applying algorithm (L: Locker pattern, R: RPC, M: Mobile agent)

| visit node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | C |
|---|---|---|---|---|---|---|---|---|---|
| paradigm | M | M | M | R | R | M | M | R | M |
| loc. of agent | 1 | 2 | 3 | 3 | 3 | 6 | 7 | 7 | C |

Table 6. After applying algorithm in reverse

Table 7 compares execution times to each other in case I. We can see that the migration pattern after applying algorithm performs better than others.

| | Only MA | Only RPC | Only locker | Algorithm | algorithm reverse |
|---|---|---|---|---|---|
| time(ms) | 3154 | 2972 | 2187 | 1769 | 2978 |

Table 7. Network execution times

The total network load of accumulated data is 115.2KB when applying algorithm, 56.2KB for applying in reverse, and 174.2KB for mobile agent only. The total network load of accumulated header is 1.3KB when applying algorithm, 0KB in case of reverse applying, and 3KB for locker pattern.

■ Case II : non-fixed node sequence to visit

The migration sequence after applying our algorithm under the assumptions above is shown in table 8. The table 9 shows migration sequence of reverse applying.

| visit node | 1 | 8 | 3 | 7 | 2 | 6 | 4 | 5 | C |
|---|---|---|---|---|---|---|---|---|---|
| paradigm | L | M | R | M | L | L | M | M | M |
| loc. Of agent | 1 | 8 | 8 | 7 | 2 | 6 | 4 | 5 | C |

Table 8. After applying algorithm

| visit node | 5 | 4 | 6 | 5 | 7 | 3 | 8 | 1 | C |
|---|---|---|---|---|---|---|---|---|---|
| paradigm | M | R | M | R | M | M | R | R | M |
| loc. of agent | 5 | 5 | 6 | 6 | 7 | 3 | 3 | 3 | C |

Table 9. After applying algorithm in reverse

Table 10 shows the comparison of execution times in the case II. From simulation results, when changing node sequence to visit by the algorithm, mobile agent consumes less execution time. But the execution time of locker pattern is little influenced, and RPC is not influenced at all.

| | Only MA | Only MA ( reverse) | Only RPC | Only locker | Algo. | algorithm reverse |
|---|---|---|---|---|---|---|
| time | 1927 | 4773 | 2972 | 2187 | 1446 | 2902 |

Table 10. network execution times[unit: ms]

In case II, the total network load of accumulated data and headers are 115.2KB, 0.37KB respectively by applying algorithm. These loads are less than those of case I. In case of mobile agent, total network load of accumulated data is about 28KB less than that of case I. If there is a big difference in the number or the size of documents at each node, the performance of system is much more influenced by node visiting sequence.

The table 11 compares the execution time of fixed node sequence with execution times of visit sequence changed by the algorithm II. From this simulation, we can see that the network execution times can be varied according to node visit sequence and interaction pattern of the paradigm as well.

| | applying algo. | Applying algo. in reverse | Only mobile agent | |
|---|---|---|---|---|
| Fixed | 1769 | 2978 | 3154 | |
| Non-fixed | 1446 | 2902 | 1927 | 4773(reverse) |

Table 11. Comparing execution times by node sequence to visit [unit: ms]

## 5. CONCLUSIONS

In this paper, we presented performance evaluation model for three different paradigms. On the basis of this model, we also presented a selection algorithm searching for more efficient interaction pattern. We then showed simulation results of these algorithms. There is no interaction pattern that is better than others in absolute terms. The choice of the interaction pattern must be performed on a case-by-case basis, according to the type of application. For performance evaluation, parameters such as CPU costs, memory usage and network traffic etc. should be considered. In this paper, we, however, concerned network traffic only, and chose a simple data mining for target application.

From this research, we can see that the network execution times are varied according to node sequence to visit and interaction pattern of the given paradigm. This paper helps us to decide convenient interaction pattern in specific application domain for developing distributed applications. In the future work, we will extend the proposed model and algorithms to facilitate non-uniform network.

## 6. REFERENCES


[1] Bic, L. F., fukuda, M., and Dillencourt, Distributed computing using autonomous objects. IEEE Computer, Aug. 1996.

[2] Wooldridge, M. and N. R. Jennings, Agent Theories, Architectures and Languages: A Survey. In Michael Jl. Wooldridge and Nicolas R.Jennings, editor, Intelligent Agent, pp.1-39, Springer-Verlag, Germany, 1995.

[3] Lange, D.B., M. Oshima, Programming and deploying Java Mobile Agents with Aglets. Addison Wesley Press, 1998.

[4] Tennenhouse, D. L., Jonathan M. S., W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. IEEE Communications, 35(1):80-86, January 1997.

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

[6] Carzaniga, A., G. P. Picco, G. Vigna, Designing Distributed Applications with Mobile Code Paradigms. Proceedings of the 19th International Conference on Software Engineering,

[7] Strasser, M., M. Schwehm, A Performance Model for Mobile Agent Systems. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'97, Volume II, pp. 1132-1140, 1997.

[8] Yariv, A., D. B. Lange, Agent Design Patterns : Elements of Agent Application Design. Second International Conference on Autonomous Agents (Agents 98), 1998.

[9] Nog, S., Chawla S., and D. Kotz, An RPC mechanism for transportable agents. Technical Report TR96-280, Department of Computer Science, Dartmouth College, Hanover, N.H., 1996.

[10] H.C.Kwon, J.H.Lee, H.J.Park, B.N.Yoon and K.J.Yoo, "A Performance Evaluation Model for Mobile Agent System. Proc. of the first International Conference on Advanced Communication Technology (ICACT'99), Muju resorts, Korea, pp.303-306 Feb. 1999.