A Performance Comparison of the Mobile Agent Model with the Client-Server Model under Security Conditions

Ki-Moon Jeong, Seung-Wan Han, Hyeong-Seok Lim
Dept. of Computer Science, Chonnam National University
300 Yongbong Dong, Buk-Ku, Kwangju, 500-757 Korea
Tel: +82-62-530-0755, Fax: +82-62-530-3439
E-mail: {u9997477, hansw, hslim}@chonnam.chonnam.ac.kr

Abstract: The traditional client-server model RPC and mobile agents have been used for interprocess communication between processes in different hosts. The performances of two mechanisms were analyzed in the literature [4, 6, 9]. But the security services which extensively affect the performance of systems because of low speed have not been considered. We propose two performance models considering the security services for the RPC and the mobile agent. Through the analysis of the models, we show that the execution time of the mobile agent takes less than the RPC in the environment considering security services.

1. Introduction

Applications in distributed computing environments have become more and more increasing recently owing to the growth of Internet usage and the progress of distributed computing technology. In such distributed applications, we have traditionally used the client-server model based on the Remote Procedure Call(RPC) for Inter-Process Communication (IPC) between processes in different hosts [1, 8]. Although this model is conceptually simple and straightforward to implement, it is not appropriate to environments in which distributed applications are executed with low-bandwidth connections or cause heavy network traffic. As the result, a mobile agent paradigm has been widely argued as an alternative one for IPC from industry and academia [3].

A mobile agent is a software that is able to autonomously migrate from one host to another in the heterogeneous network, to perform some computation on behalf of the owner. Mobile agents have two main characteristics: mobility and autonomy. Owing to these characteristics, they may reduce network traffic, increase asynchrony between clients and servers, and add client-specified functionality to servers.

In many recent papers, the authors discussed about the power of the mobile agent paradigm against traditional communication paradigms and argued that it is useful for many applications such as information retrieval, network management, mobile computing and so on. Moreover several researchers proposed each performance model for IPC paradigms and analyzed the proposed models quantitatively [2, 4, 6, 9]. In order to compare two paradigms, they considered some parameters that affect network load and execution time. But they did not exactly reflect real distributed computing environments because the more parameters must be considered.

This work was partially supported by the Korea Science and Engineering Foundation under contract 98-0102-11-01-3, and partially by the Brain Korea 21 Project.

Real distributed environments are vulnerable to a variety of attacks: masquerade, information disclosure, integrity violation and so forth [5]. So as to execute real applications securely in distributed computing environments, suitable measures against these attacks must be taken. Because the operations used to provide the security services are much slower than other operations, security services affect the performance of systems extensively. However, the models proposed until now for evaluating IPC paradigms did not include the factors that appear when security services are considered in distributed computing environments. Therefore, in order to exactly analyze the performance for IPC paradigms as in real environments, new model in which the parameters related to security are included must be designed. In this paper, we propose two new performance models in which the equivalent security services are considered in RPC and mobile agent, and analyze the performance of these models.

This paper is organized as follows: In Section 2, several performance models for IPC paradigms are examined. Section 3 presents security services that must be considered. In Section 4, we propose the Petri net model and the performance formulas for IPC paradigms under security conditions. Finally, Section 5 concludes this paper.

2. Related work

In [9], the performance models of RPC and mobile agent using formulas are presented. It introduced a single interaction performance model and a sequential interaction performance model derived from the former for evaluating the performance of the application made by given scenario. The performance of IPC paradigms is affected by application selectivity of the agent and the size of reply message. Consequently, the paper presented that RPC and mobile agent are used alternately better than a pure RPC or mobile agent.

Several distributed mechanisms (RPC, Remote Evaluation (REV), and mobile agent) are modeled in [6] using Petri net and analyzed through the WebSPN tool. In [6], the mobile agent does not provide optimal performance because it is dependent on external factor such as network speed and characteristic of application. But mobile agent is regarded as a suitable mechanism, when the communication subsystem is fast enough or in all the cases when the user may be temporarily disconnected from the system due to mobility or network unavailability.

A performance evaluation of the mobile agent paradigm in comparison to the client-server paradigm is presented in [4]. The evaluation was conducted on the Java environment. The result showed that significant performance improvements can be obtained using mobile agent.

Those papers only considered the parameters affecting the performance such as network speed and application selectivity of the agent. They did not consider security services which are required parameters for executing applications securely and exactly in the distributed computing environments.

3. Security services

Distributed computing environments are very vulnerable to included entities. In such environments, there are many attacks like masquerade, information disclosure, integrity violation and so on. There are five main security services in this environment. Authentication service, access control service, confidentiality service, integrity service, and audit service. In order to provide an analysis of the performance offered by the RPC and the mobile agent paradigm under the circumstances with security services, we consider the followings.

We assume that there is no denial of service attack and programs are executed correctly. Among security services, authentication service, access control service, and audit service are independent of evaluating the performance of IPC paradigms because they are not directly related with communication in hosts. In the case of authentication service, because both the RPC and the mobile agent model authenticate users, hosts, and process only once, we can assume that they have the same cost in network load and execution time without losing generality. In access control service and audit service, because access control service is implemented in local host to protect its own resource from any illegal access and audit service to record the security events for finding a violator via a post-mortem examination, we can treat them like authentication service.

However, confidentiality service and integrity service are different from the above-mentioned services. When hosts exchange the message with each other, it must be encrypted and digitally signed by using cryptographic mechanisms in order to be protected from being disclosed to unauthorized entities and being changed without authorization. In general, cryptographic mechanisms require the operations having very high cost and their execution time is directly proportional to the amount of data processing. Consequently, the using frequency of such mechanisms and the amount of data processing have quite an effect on the performance of distributed system.

In order to provide the equivalent security services for two paradigms, the RPC only needs confidentiality and integrity of request and reply message, but the mobile agent requires additional consideration. The mobile agent consists of program codes, state of agent, and data of execution result. In the mobile agent paradigm we consider two main issues. First, protect the host against hostile action of malicious mobile agent. Second, protect the mobile agent against tampering attempts by the executing host[5]. Many techniques such as proper access control resources of host for the first problem have been developed. But second problem seems to be much harder because malicious host can analyze code, so see the program source and state of the mobile agent, then illegally change the data of the agent [7]. In order to protect the mobile agent, every host must verify the owner's signature for integrity of program code and encrypt, decrypt, sign, and verify the state of agent and the data of agent for confidentiality and integrity whenever agent migrates from one host to another. Nevertheless, it is very difficult to provide the absolute confidentiality of program code until now.

We assumed that all entities are users of a Public Key Infrastructure(PKI) for providing the cryptography mechanism such as encryption, decryption, signature, and verification and so on.

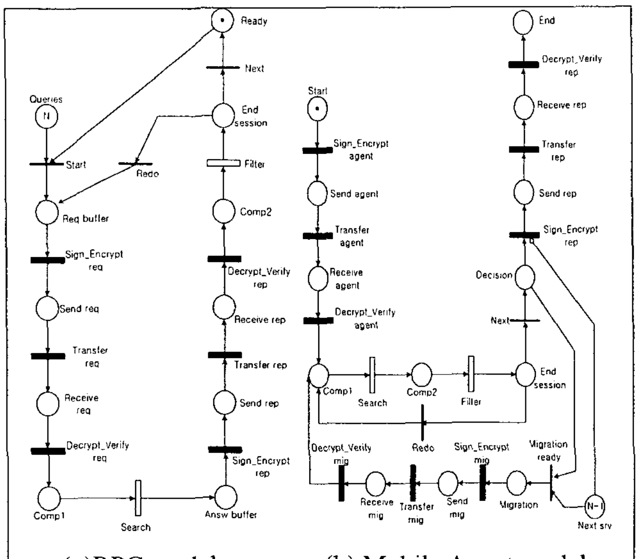
4. Performance model

4.1 Modeling of IPC paradigms

In this section we describe two Petri net models in which the above-mentioned security services are applied to the RPC and the mobile agent system. We assume that proposed models execute information retrieval, involving a set of N DB-servers. All servers have to be contacted, and some of them more than once.

In Figure 1(a), a Petri net model for RPC is shown. The process of operations is described simply as follows. At the beginning, place *queries* contains N tokens, which represents the N requests that client has to send to servers. Also *ready* contains a token, indicating that a request is ready to be processed. Before a client sends the request to server, request is signed by private key of the client and encrypted by public key of the server (transition *Sign_Encrypt req*). The request signed and encrypted is transferred to DB-server, then the request is decrypted by private key of the server and verified by public key of client for validity of the request (transition *Decrypt_Verify req*). If the request is valid, server executes search. Then the result of searching is signed by private key of the server and encrypted by public key of the client in order to be transferred to client (transition *Sign_Encrypt rep*). The transferred reply is decrypted by private key of the client and verified by public key of the server for validity of the reply (transition *Decrypt_verify rep*). Then the reply is refined for client (transition *Filter*). When a token enters *End_session*, the next process should be decided whether it sends a request to same server (firing of *Redo*) or closes the current session (firing of *Next*) for sending the request to next server or ending the job. This procedure is iterated until the condition that *Queries* contains none and *Ready* contains one token.

In Figure 1(b), the model of a mobile agent system is shown. A token in *Start* denotes the client in a ready state waiting to start the session. When the agent transfers to DB-server for information retrieval through the *Sign_Encrypt agent*, the code of agent and initial state of agent are signed by owner's private key and encrypted by public key of the destination server. The transferred agent to server is decrypted by private key of the server and verified by owner's public key for validity of message (transition *Decrypt_Verify agent*). Then the agent executes the information retrieval and refinement of raw information in server. In the *End session*, the server has to decide whether sends the result of execution on the server to another server (transition *Next*) or executes information retrieval on the current server again (transition *Redo*). If *Next* fires, the server should decide whether sends the result to client (firing of *Sign_Encrypt rep*) or sends the result and the code to another server (firing of *Migration ready*). In *Sign_Encrypt rep*, the result is signed by private key of the server and encrypted by public key of the client, and transferred to client. Then the client decrypts and verifies reply for validity of the reply (transition *Decrypt_Verify rep*). When the migration is decided, current server signs and encrypts current state of the agent and result using private key of the current server and public key of the destination server respectively then transfers this message and program code of the agent to next server. The transferred message is decrypted by private key of the current server and verified by public key of the previous server, and the signature of code is verified by public key of owner of agent.



(a)RPC model (b) Mobile Agent model

Figure 1. Performance models considered security services using Petri net

In order to analyze properly the RPC and the mobile agent models, we use the parameters considering the security services as follows:

- D_{req} : a constant representing the size of request.
- D_{rep} : the size of reply with uniform distribution.
- D_{code} : a constant representing the size of the code of a mobile agent.
- D_{state} : a constant representing the size of the state of a mobile agent.
- D_{data} : the size of the data of the mobile agent. This is made by server after searching. We consider this size as $D_{rep}(1-\sigma)$ $\{\sigma \mid$ selectivity of the agent $(0 \leq \sigma \leq 1)$ $\}$, because reply size for client is reduced by selectivity of agent compare with RPC. We assume that this size have uniform distribution.
- R_{se} : throughput for searching the data in the server. The value is exponential distributed.
- R_{rf} : throughput of data processing for refining. This processing can take place in the server and in the client, depending on the model. The value is exponential distribution.
- R_s : throughput of the signature.
- R_v : throughput of the verification.
- R_e : throughput of the encryption.
- R_d : throughput of the decryption.
- R_{th} : throughput of the communication network.

We assume that the size of signature and the padding size of encryption are zero. So we do not include those in the set of parameters. The values assigned to the timed transitions of the Petri net models are calculated according to the formulas shown in table 1.

CS (RPC) model		
Transition	Type	Expression
Sign_Encrypt req	Deterministic	$\frac{D_{req}}{R_s} + \frac{D_{req}}{R_e}$
Transfer req	Deterministic	$\frac{D_{req}}{R_{th}}$
Decrypt_Verify req	Deterministic	$\frac{D_{req}}{R_d} + \frac{D_{req}}{R_v}$
Search	Exponential	R_{se}
Sign_Encrypt rep	Uniform	$\frac{D_{rep}}{R_s} + \frac{D_{rep}}{R_e}$
Transfer rep	Uniform	$\frac{D_{rep}}{R_{th}}$
Decrypt_Verify rep	Uniform	$\frac{D_{rep}}{R_d} + \frac{D_{rep}}{R_v}$
Filter	Exponential	R_{rf}
MA model		
Sign_Encrypt agent	Deterministic	$\frac{(D_{code} + D_{state})}{R_s} + \frac{(D_{code} + D_{state})}{R_e}$
Transfer agent	Deterministic	$\frac{D_{code} + D_{state}}{R_{th}}$
Decrypt_Verify agent	Deterministic	$\frac{(D_{code} + D_{state})}{R_d} + \frac{(D_{code} + D_{state})}{R_v}$
Search	Exponential	R_{se}
Filter	Exponential	R_{rf}
Sign_Encrytp rep	Uniform	$\frac{D_{data}}{R_s} + \frac{D_{data}}{R_e}$
Transfer rep	Uniform	$\frac{D_{data}}{R_{th}}$
Decrypt_Verify rep	Uniform	$\frac{D_{data}}{R_d} + \frac{D_{data}}{R_v}$
Sign_Encrypt mig	Uniform	$\frac{D_{data}+D_{state}+D_{code}}{R_s} + \frac{D_{data}+D_{state}+D_{code}}{R_e}$
Transfer mig	Uniform	$\frac{D_{data} + D_{state} + D_{code}}{R_{th}}$
Decrypt_Verify mig	Uniform	$\frac{D_{data}+D_{state}+D_{code}}{R_d} + \frac{D_{data}+D_{state}+D_{code}}{R_v}$

Table 1. Formulas for Petri net models

We describe the RPC model and the mobile agent system considering the security services as formulas. We assume that a client requires the processing by all of the N servers(the job is not ended before contacting all of the servers). All the servers have to be visted, some of them more than once. Also we assume that the time of marshalling, unmarshalling, and network delay is zero. The parameter p is the probability of using the same server for the subsequent processing (the probability of firing transition *Redo*).

First, we present the client-server(RPC) model. The network load L_{rpc} consists of the size of the request and the size of the reply. The request and the reply are repeated more than N.

$$L_{rpc} = N\left(\frac{1}{1-p}\right)\left(D_{req} + D_{rep}\right)$$

The execution time consists of the time for searching and filtering the data and the time for providing the security services. T_{rpcth} denotes the time for searching and filtering the data and T_{rpcses} denotes the time for providing the security services.

$$T_{rpcth} = \frac{D_{req}}{R_{th}} + \frac{D_{rep}}{R_{th}} + \frac{1}{R_{se}} + \frac{1}{R_{rf}}$$

$$T_{rpcses} = \frac{D_{req}}{R_s} + \frac{D_{req}}{R_e} + \frac{D_{req}}{R_d} + \frac{D_{req}}{R_v} + \frac{D_{rep}}{R_s} + \frac{D_{rep}}{R_e} + \frac{D_{rep}}{R_d} + \frac{D_{rep}}{R_v}$$

$$T_{rpc} = N\left(\frac{1}{1-p}\right)\left(T_{rpcth} + T_{rpcses}\right)$$

Second, we present the mobile agent system. When the mobile agent begin the job the client sends program code of agent and initial state of the agent to server so the initial data $D_{init} = D_{code} + D_{state}$. When the agent migrates between servers agent additionally includes the data produced by server, so the migration data $D_{mig} = D_{code} + D_{state} + D_{data}$.

The network load L_{ma} consists of three parts. The first part is the initial data of the agent when the job is started. The second part is the migration data. Whenever the agent migrates *N-1* servers the agent includes D_{mig}. The last part is the reply data. The last server sends a client the reply which only includes the result of searching and filtering. So network load as follows.

$$L_{ma} = D_{init} + D_{data} + D_{mig}(N-1)$$

The composition of the execution time is the same as the RPC. T_{math} denotes the time for searching and filtering the data. The time for providing the security services consists of three parts. The first part is the time required for signature, encryption, and decryption for D_{init} and the time required for verification of code's signature for integrity of agent's code. The second part is the time needed encryption, decryption, signature, and verification for confidentiality and integrity of data occurred from *N-1* servers. The third part is the time required for security service to reply.

$$T_{math} = \frac{D_{init}}{R_{th}} + \frac{D_{data}}{R_{th}} + (N-1)\left\{\left(\frac{1}{R_{se}} + \frac{1}{R_{rf}}\right)\left(\frac{1}{1-p}\right) + \frac{D_{mig}}{R_{th}}\right\}$$

$$T_{mases} = \frac{D_{init}}{R_s} + \frac{D_{init}}{R_e} + \frac{D_{init}}{R_d} + N\frac{D_{code}}{R_v} + \frac{D_{data}}{R_s} + \frac{D_{data}}{R_e} + \frac{D_{data}}{R_d} + \frac{D_{data}}{R_v} +$$

$$(N-1)\left\{\frac{(D_{state}+D_{data})}{R_s} + \frac{(D_{state}+D_{data})}{R_e} + \frac{(D_{state}+D_{data})}{R_d} + \frac{(D_{state}+D_{data})}{R_v}\right\}$$

$$T_{ma} = T_{math} + T_{masc}$$

4.2 Analysis of Performance

The performance models are analyzed using the formulas described in previous section. Some of the numerical values for the parameters used formulas are referred to [6]. We assume that cryptography algorithms are RSA algorithm and DSA algorithm. RSA is used for encryption and decryption, and DSA is used for signature and verification. We get the values of cryptography parameters after testing. We use the cryptography algorithms made by JAVA, 1024-bit key and 100kbyte message on the Intel PentiumⅢ/500MHz. The numerical values are listed in Table 2.

Parameter	Mean	Value
N	Num. of client request	10
D_{req}	Size of the request	1kB
D_{rep}	Size of the reply	(1kB, 30kB)
D_{code}	Size of the agent code	39kB
D_{state}	Size of the agent state	4kB
R_{se}	Search rate	4req/s
R_{rf}	Refine rate	4req/s
R_{th}	Network throughput	1000kbps
R_e	Encryption throughput	533.33kbps
R_d	Decryption throughput	14.55kbps
R_s	Signature throughput	22.95kbps
R_v	Verification throughput	8000kbps

Tabel 2. Numerical values for parameters

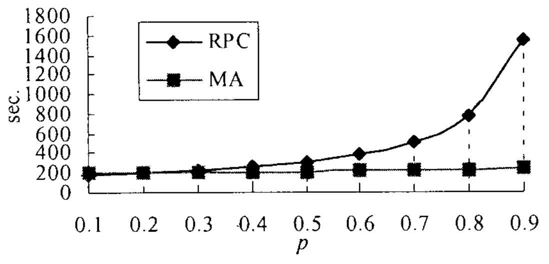


Figure 2. Execution time versus p for the $\sigma = 0.1$

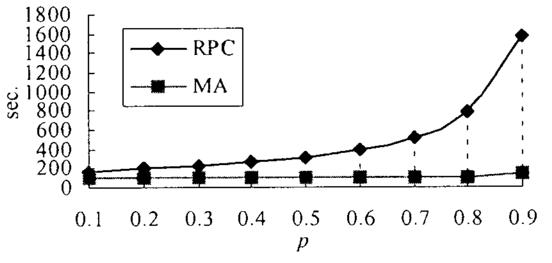


Figure 3. Execution time versus p for the $\sigma = 0.9$

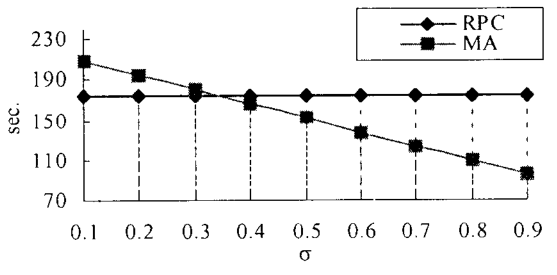


Figure 4. Execution time versus σ for the $p = 0.1$

Figure 2 and Figure 3 show graphs that describe the execution time according to p for $\sigma = 0.1$ and $\sigma = 0.9$ respectively. The σ indicates the selectivity of the agent, so the load of network decreases according as the value of σ increases. When the value of σ is 0.1, the performance of the mobile agent is better than the RPC in $p>0.2$. The execution time of the mobile agent takes always less than the RPC in $\sigma = 0.9$. In the IPC paradigm considering security services, the execution time of the mobile agent takes nearly less than the RPC. Moreover the RPC is sudden increase according to the characteristic of the application (the increase of the p). But the mobile agent maintains the execution time invariably without sudden increasing. Although the feature of the applications(the p) is considered the mobile agent's performance is better than the RPC's performance in most conditions.

Figure 4 shows the comparison the execution time of the RPC with the execution time of the mobile agent for a fixed probability of $p = 0.1$ while varying the selectivity σ between 0.1 and 0.9. The σ do not affect the RPC. But the mobile agent executes the refinement of the searched data not in the client but in server. In the result, the size of the reply for the client is smaller than the size of the RPC's reply.

5. Conclusion

In order to execute an application the user should use security services in distributed computing environments because distributed computing environments are very vulnerable to a variety of attacks. We propose new performance models considering security services for the RPC and the mobile agent using the Petri net and formulas. Also, we analyze the performance.

The security services require very high cost. In the distributed computing environment the cost of security services increases according to the network load. If the application has heavy network load, the mobile agent is faster than the RPC in the proposed models.

The mobile agent requires the more security services than services that we considered in real environment. It is very difficult to protect a mobile agent from a malicious host until now. So there would be much cost to perfectly protect the mobile agent. This issue is not addressed in this paper.

Although the mobile agent has the above disadvantage it has the merits such as mobility and autonomy. Because of such characteristics the mobile agent can communicate with hosts without many interactions. This is the improvement of the performance in the distributed environment where the distributed application and the mobile computing are increasing.

References

[1] A.D. Birrell, B.J. Nelson, "Implementing remote procedure calls," ACM56 Transactions on Computer Systems 2(1), pp.39-59, 1984.

[2] Tack-How chia, Srikanth Kannapan," "Strategically Mobile Agents," Proceedings of First International workshop on Mobile Agents'97, pp.149-161, April 1997.

[3] C.G. Harrison, D.M. Chess, A. Kershenbaum, "Mobile Agents: Are they a good idea?," Research Report, IBM Research Division T.J. Watson Research Center, March 1995.

[4] L. Ismail, D. Hagimont, "A Performance Evaluation of the Mobile Agent Paradigm," Proceedings of the 1999 ACM SIGPLAN conference on Object-oriented Programming, systems, languages, and applications, pp.306-313, November 1999.

[5] Wayne Jansen, Tom Karygiannis, "Mobile Agent Security," NIST Special Publication 800-19, 1999.

[6] A. Puliafto, S. Riccobene, M. Scarpa, "An analytical comparison of the client-serer, remote evaluation and mobile agents paradigms," Proceedings of First International Symposium on Agents Systems and Applications, pp.248-292, October 1999.

[7] Tomas Sander, Christian F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts," in G. vigna (Ed.), Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.

[8] James W. Stamos, David K. Gifford, "Remote Evaluation," ACM Transactions on Programming Languages and Systems, Vol.12, No.4, pp.537-565, October 1990.

[9] Markus Straßer, Markus Schwehm, "A Performance Model for Mobile Agent Systems," Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), Vol.II, CSREA, pp.1132-1140, 1197.