Using Adaptive Agents for the Fault-Tolerant Mobile Computing System

Taesoon Park, Jaehwan Youn, and Dongryung Kim

Department of Computer Engineering, Sejong University,
Seoul 143-747, Korea
tspark@sejong.ac.kr, {rivian, fishangel}@sju.ac.kr

Abstract. This paper presents a fault-tolerance scheme based on mobile agents for the reliable mobile computing systems. The mobility of the agent is suitable to trace the mobile hosts in the system and the intelligence of the agent makes it efficient to support the fault-tolerance services. The proposed scheme especially focuses on the adaptiveness of the agent. The agents try to keep the failure-recovery cost and the failure-free operation cost within a certain level, regardless of the hand-off frequency of the mobile hosts. They also try to balance the two costs.

1 Introduction

Fault-tolerance is an important design issue to build a reliable mobile computing system and many fault-tolerance schemes have been proposed [2, 3, 5]. However, the fault-tolerance of these schemes is mainly supported by the mobile support stations of the system and hence these schemes have the following problems in common: First, the information required for the recovery of mobile hosts is managed by the mobile support station, which may cause the high processing delay of the support station. Second, for the efficiency of the recovery, the information managed by one support station is transferred to the new support station as the corresponding mobile host moves to the new area, which may cause the severe hand-off delay. Third, if the recovery information is not migrated to reduce the hand-off delay, the mobile host experiences a longer recovery delay in case of a failure.

To solve these problems, we have suggested a mobile agent based fault-tolerance scheme for the mobile computing system [4]. In this scheme, a stationary agent residing in each mobile support station site takes care of the recovery information of mobile hosts so that the support station can concentrate on its own tasks, such as the mobility handling and the communication of the hosts. Also, a set of mobile agents for each mobile host take care of the migration of the recovery information and hence the recovery information migration of a host can be performed asynchronously with its hand-off. As a result, the fault-tolerance service does not cause any unnecessary hand-off delay. Another notable point of this scheme is that mobile agents can make a migration decision suitable for the behavior of each mobile host.

However, in [4], the structure of the cooperative agents servicing the fault-tolerance and their interaction models are mainly discussed and not much attention is given to the migration strategy of the mobile agent carrying the recovery information. For the efficient fault-tolerance service, the migration decision of the agent is very important since the frequent migration may cause the severe failure-free operation cost and the lazy migration may cause the longer recovery delay of the host. Hence, in this paper, a new migration strategy of the mobile agent carrying the recovery information is proposed. The proposed scheme employs the adaptive agent which keeps evaluating the possible failure-recovery cost and the failure-free operation cost as the behavior of the mobile host, such as the hand-off frequency or the communication frequency, varies. Based on the evaluation, the agent schedules its migration to balance the two costs within a certain range.

2 Background

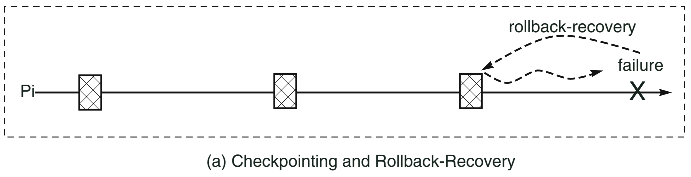
2.1 Mobile Computing System

A *mobile host (MH)* is an entity which executes the computation and communicates with another MH while traversing the system. In order to provide efficient and seamless services for MHs, the entire area covered by the mobile computing system [1] is divided into a number of service areas called the *cells* and one *mobile support station (MSS)* per cell is employed. Main tasks of the MSS are the location management and the seamless communication services for the MHs. For these services, MSSs are connected with each other through the high speed wired network; and when an MH leaves a cell and enters the next cell, two corresponding MSSs exchange information regarding the location and the communication status of the MH. This procedure is called the *hand-off.* The computation performed by the processes of an MH is assumed to follow a *piece-wise deterministic model*, in which a process can always produce the same sequence of states for the same sequence of message-receiving events.

2.2 Checkpointing and Message Logging

Checkpointing is an operation to save the intermediate states of the processes into the stable storage, so that the processes can restore the saved states and resume the computation from the restored states when the system fails [3]. Figure 1.(a) shows the effect of checkpointing. The horizontal arrow and the shaded boxes in the figure denote the progress of a process, P_i, and its periodic checkpointing, respectively. When P_i fails, it can resume the computation from the latest checkpoint, instead of restarting from the initial state, which is called the *rollback-recovery.*

Message logging is the operation to save the incoming messages of a process into the stable storage so that the process can re-execute the computation with the logged messages and reproduce the exactly same states which have occurred right before the failure [3]. Figure 1.(b) shows the effect of message logging. The



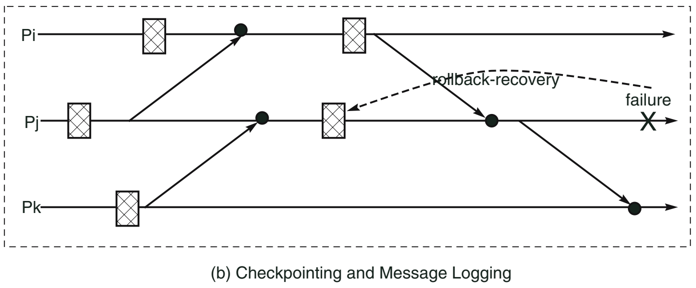


Fig. 1. Checkpointing, Message Logging and Rollback-Recovery

small black circle in Figure 1.(b) denotes the logging of each incoming message. With the message logging, P_j can re-process the logged messages during the recomputation, instead of receiving new messages, so that it can produce the same computational states which have been occurred right before the failure. As a result, the failure of P_j does not affect the other processes.

When the checkpointing and the message logging are considered for the mobile computing system, MSSs provide the stable storage since the storages of MHs are very limited. However, the MH moves around the system and hence checkpoints and message logs saved for an MH become dispersed as the MH moves over a number of cells. To efficiently manage the checkpoint and the message log, many approaches have been proposed [5].

3 Fault-Tolerance Service Based on Adaptive Agents

3.1 Basic Scheme

One common problem of existing schemes is to manage the checkpoint and the message logs as one pack. As shown in Figure 1, when a process fails, it first restores its latest checkpoint and then re-processes the logged messages. Hence, the checkpoint near the MH can help the early start of the recovery however the logged messages would be used gradually as the recomputation proceeds. Considering the heavy migration cost of the checkpoint and message log pack, it is wiser to separately manage these two types of recovery information. We have presented a basic design of mobile agent based recovery information management scheme in [4].

In the proposed design, three types of agents are used: One is the stationary agent, called a *Recovery Agent (RA)*. For each MSS site, one RA is used to temporarily manage checkpoints and message logs produced by the MHs visiting the cell. The others are mobile agents, called the *checkpointing agent (CPA)* and the *log agent (LGA)*, each of which manages and makes the migration decision of the latest checkpoint and the message logs, respectively. The CPA retrieves the latest checkpoint from the RA and migrates to the local MSS site of the corresponding MH. The LGA retrieves the message logs from the RA or it can just record the log location to relieve the large burden of the log migration. Note that the migration of CPA and LGA is asynchronous with the migration of the corresponding MH.

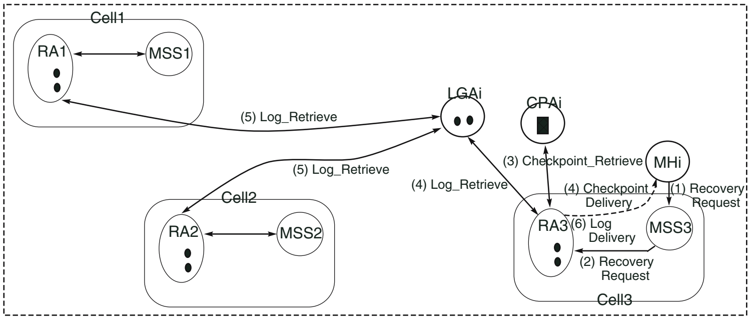


Fig. 2. Recovery based on Cooperative Agents

When an MH fails, the agents cooperate with each other to recover the failed MH. Figure 2 describes the recovery process taken by these agents. When MH_i in $Cell_3$ fails, the recovery agent, RA_3, takes the prime responsibility of MH_i's recovery. Hence, on the receipt of the *recovery* message from MH_i, RA_3, contacts with CPA_i and LGA_i to retrieve the latest checkpoint and the logged messages for MH_i. On the request from RA_3, CPA_i replies with the latest checkpoint and LGA_i sends the *log_collection* messages to the recovery agents recorded in its visiting list. After collecting the logged messages from other recovery agents, LGA_i, forwards the logs to RA_3. For the early start of the recovery process, RA_3 starts the recovery of MH_i as soon as it receives the latest checkpoint from CPA_i, without waiting for the completion of the log collection. Then, RA_3 sorts the logged messages in the message sequence order and sends them to MH_i.

3.2 Adaptive Agent Based Fault Tolerance Service

One advantage of using mobile agents for the fault-tolerance service is that it is possible to design the efficient and proper agent migration strategy for each MH. Owing to a variety of hand-off rates, checkpointing intervals and communication rates of the MHs, the uniform migration strategy applied to all the MHs may result in the unpredictable and inefficient performance. Hence, to maintain the overhead caused by CPA and LGA migration to be in a predictable level, the careful design of the agent migration strategy suitable for each MH is desirable.

• **Distance-based scheme:** One important performance measure of this fault-tolerance service is the failure-recovery cost, which is the cost to retrieve the latest checkpoint and the message logs. The expected failure-recovery cost of MH_i, say C^i_{FR}, can be expressed as

$$C^i_{FR} = \alpha * D^i_{CPA} * S^i_{CPA} * C_N + \beta * \Sigma_{j=1}^{N_{log}} D_{log_j} * S_{log_j} * C_N, \qquad (1)$$

where D^i_{CPA} is the expected distance between MH_i and its CPA at the time of a failure; S^i_{CPA} is the expected size of the CPA; C_N is the network cost for an object of the unit size to move in the unit distance; $D^i_{log_j}$ and $S^i_{log_j}$ are the expected migration distance and the expected size of each message log of MH_i; and N_{log} is the expected number of logged messages to be migrated for each failure-recovery. The values of α and β are the weights. Since for the fast recovery, it is more important to retrieve the latest checkpoint than to collect the message logs, as it is discussed before, more weights should be put on the checkpoint retrieval.

Since our focus is not to analyze the performance but to select the major factors to affect the performance, we can simplify the failure-recovery cost as the function of D^i_{CPA}, such as

$$C^i_{FR} = k * D^i_{CPA}, \qquad (2)$$

where $k = S^i_{CPA} * C_N$ and it is assumed that the values of S^i_{CPA} and C_N are constant throughout the execution of MH_i; and all the weights are put on the checkpoint retrieval, such that $\alpha = 1$ and $\beta = 0$. Now, the system can maintain the constant or predictable failure-recovery cost for MH_i by adjusting the value of D^i_{CPA}. We call this scheme the *distance-based scheme* for the CPA migration. To implement the distance-based scheme, a proper value of D_{CPA} is first selected for the wanted level of C_{FR}. When MH_i migrates, it increments its migration counter; and it asks the RA for the migration of the CPA when the counter reaches the predetermined value.

• **Time-based scheme:** Another important performance measure of this service is the failure-free cost, which is the cost to migrate the CPA and the LGA (with or without its message logs) during the failure-free operation of the MH. Let C^i_{FF} be the expected failure-free cost of MH_i per unit time. Then, C^i_{FF} can be obtained as

$$C^i_{FF} = \frac{1}{T^i_{CPA}} * D^{i'}_{CPA} * S^i_{CPA} * C_N + \frac{1}{T^i_{LGA}} * D^{i'}_{LGA} * S^i_{LGA} * C_N, \qquad (3)$$

where T^i_{CPA} and T^i_{LGA} are the time intervals between two consecutive migration events of the CPA and the LGA, respectively; and $D^{i'}_{CPA}$ and $D^{i'}_{LGA}$ are the expected distances between MH_i and its CPA or LGA for each migration, respectively. To simplify the equation, it is assumed that the values of S^i_{CPA}, S^i_{LGA} and C_N are constant throughout the execution of MH_i; and we also assume the case that the LGA does not carry any message log. Under this assumption,

$S^i_{LGA} << S^i_{CPA}$ and hence the second term is ignored. Then, the failure-free cost can be rewritten as

$$C^i_{FF} = k' * \frac{1}{T^i_{CPA}} * f(T^i_{CPA}), \tag{4}$$

where $k' = S^i_{CPA} * C_N$ and $D^{i'}_{CPA} = f(T^i_{CPA})$. Note that the value of D_{CPA} is expressed as a function of T_{CPA} since the longer migration interval allows the more hand-offs of a MH.

Now, to maintain the constant and predictable failure-free cost regardless of the various behaviors of MHs, the *time-based scheme* for the CPA migration is suggested. In the time-based scheme, a proper time-out value, T_{CPA} is first selected for the wanted level of C_{FF}. The CPA queries the current location of MH_i and migrates to the corresponding MSS site, whenever the time-out timer expires. When the CPA moves, it calculates the moving distance and sets the next timer value as the value to $T_{old} * \frac{D_{old}}{D_{new}}$, where T_{old} is the previous time-out value, D_{new} and D_{old} are the current and the previous moving distances. As a result, the constant C_{FF} value can be managed.

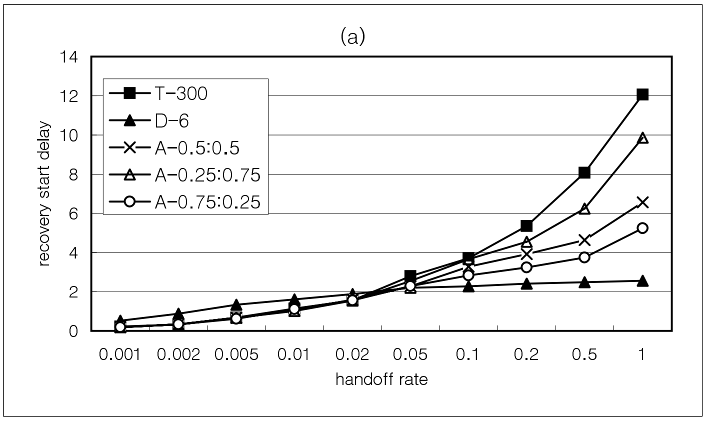
• **Adaptive scheme:** However, when both of the failure-recovery cost and the failure-free cost are considered together, it is easily noticed that the two notions of time-based and distance-based schemes conflict with each other. When the short value of D_{CPA} is selected to reduce the C_{FR}, the value of $\frac{1}{T_{CPA}}$ should become larger, which causes the higher C_{FF} value; and vice versa. Hence, when the system wants to have low costs for both of the failure-recovery and the failure-free operation, the CPA should take one performance measure ignoring the other; or it should take a certain level of the costs in-between two wanted values.

For example, suppose that MH_i wants the value of D^i_{CPA} to be d and the value of T^i_{CPA} to be t. Let $C^i_{FR}[d]$ and $C^i_{FF}[d]$ be the failure-recovery cost and the failure-free cost when the distance-based scheme with the distance value of d is applied for the CPA migration. Similarly, let $C^i_{FR}[t]$ and $C^i_{FF}[t]$ be the costs when the time-based scheme with the time value of t is applied. Then, the adaptive CPA tries to keep the failure-recovery cost to be $\alpha * C^i_{FR}[d] + \beta * C^i_{FR}[t]$ and the failure-free cost to be $\beta * C^i_{FF}[d] + \alpha * C^i_{FF}[t]$, where $\alpha + \beta = 1$. Note that when $\alpha = 1$, the CPA chooses the constant failure-recovery cost; and when $\beta = 1$, the CPA chooses the constant failure-free cost. Otherwise, the CPA chooses the costs in-between the wanted ones.

To implement this adaptive migration strategy, the CPA basically follows the time-based method and the MH assigns the wanted values of T_{CPA}, D_{CPA}, α and β. The adaptive CPA first performs the time based migration with the given time-out value of T_{CPA}. For the first k migration, it observes the expected distance between the CPA and the MH, say D^E_{CPA}, under the given T_{CPA} and sets the target level of the distance as $\alpha * D_{CPA} + \beta * D^E_{CPA}$. Then, for each migration, the CPA examines the migration distance. If the migration distance exceeds the target distance, it decreases the value of T_{CPA} to reach the target distance. Since the adaptive CPA keeps adjusting the migration frequency considering the expected recovery cost, the overall performance can be maintained within the expected level.

4 Performance Study

To measure the performance of the adaptive agent based fault-tolerance service, a simulation study has been performed. We have simulated a mobile computing system consisting of 100 mobile hosts and 1000 X 1000 rectangular cells [1]. One mobile support station is assumed for each cell. A mobile host stays in a cell for $\frac{1}{\lambda_h}$ time units and then migrates to one of its eight neighbor cells. The host communicates with another mobile host and the communication rate of each mobile host is $\lambda_m = 1/10$. The checkpointing rate and the failure rate of each mobile host are assumed to be $\lambda_c = 1/1000$ and $\lambda_f = 1/10000$. These rates are assumed to follow the exponential distribution.



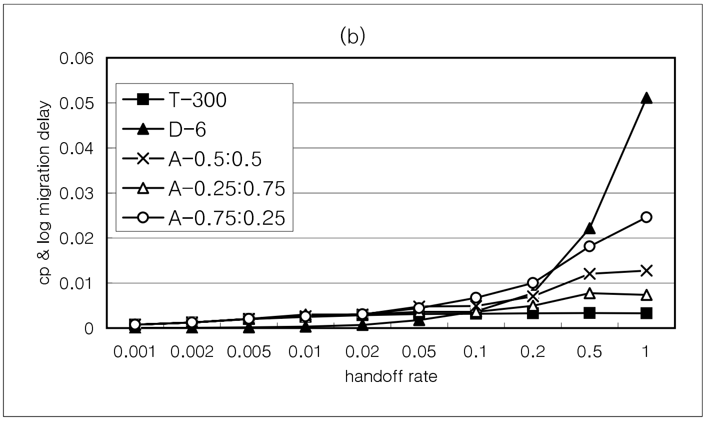


Fig. 3. Performance of the Adaptive Agents

Figure 3.(a) and (b) show the failure-recovery cost and the failure-free cost of the schemes, respectively. As it is expected, the distance based scheme with the distance value of 6, which is denoted by *D-6*, produces the constant failure-recovery cost while its failure-free cost is unpredictable. Also, the time based scheme with the time value of 300, which is denoted by *T-300*, guarantees the constant failure-free cost while its failure-recovery cost can be too high. The performance of adaptive schemes with different weights are denoted by $A\text{-}\alpha{:}\beta$ in the figure. The performance results show that the adaptive CPA effectively adjusts its performance for the various hand-off rates.

5 Conclusions

In this paper, we have presented the adaptive agent based fault-tolerance scheme for the reliable mobile computing system. The proposed scheme especially focuses on the adaptiveness of the agent which keeps the failure-recovery cost and the failure-free operation cost within a certain level considering the importance of the two performance measures. We also presented the simulation results supporting the effectiveness of the adaptive agents.

Acknowledgments

This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD)(KRF-2005-041-D00745) and also supported by grant No. B1220-0401-0037 from the University Fundamental Research Program of Ministry of Information & Communication in Republic of Korea.

References

1. Ayildiz, I.F., Ho, J.S.M.: On Location Management for Personal Communications Networks. IEEE Communications Magazine (1996) 138–145
2. Neves, N., Fuchs, W.K.: Adaptive Recovery for Mobile Environments. Communications of the ACM, Vol. 40, No. 1 (1997) 68–74
3. Park, T., Woo, N., Yeom, H.Y.: An Efficient Optimistic Message Logging Scheme for Recoverable Mobile Computing Systems. IEEE Transactions on Mobile Computing, Vol. 1, No. 4 (2002) 265–277
4. Park, T.: Mobile Agent based Fault-Tolerance Support for the Reliable Mobile Computing Systems. Lecture Notes in Computer Science, Vol. 3454 (2005) 173–187
5. Pradhan, D.K., Krishna, P., Vaiday, N.H.: Recoverable Mobile Environment : Design and Trade-Off Analysis. Proc. of the 26th Int'l Symp. on Fault Tolerant Computing Systems (1996) 16–25