マルチメディア通信と分散処理 80－1
グループウェア 21－1
（1997. 1. 30）

Hybrid Checkpointing in Mobile Computing System

Takeaki Yoshida, Hiroaki Higaki, and Makoto Takizawa
Tokyo Denki University
e-mail {take, hig, taki}@takilab.k.dendai.ac.jp

Mobile stations support neither enough volume of storage and processing power nor enough capacity of battery to do reliable, long-term communications. Hence, the communication channels with the mobile stations are often disconnected. It is important to realize the fault-tolerant computation among the mobile stations. It is difficult for multiple mobile stations to take synchronous checkpoints since the communication links with the mobile stations may be disconnected even during taking the checkpoints. In this paper, we discuss a hybrid checkpoint, where checkpoints are taken asynchronously by the mobile stations and synchronously by the fixed stations.

移動体環境のための複合型チェックポイント

吉田 丈成　桧垣 博章　滝沢 誠
東京電機大学理工学部経営工学科

移動体端末は、十分な通信と処理能力を有していない。分散型応用を高信頼なものとするために、チェックポイントを取る方法がある。これまでに、いくつかの同期型チェックポイント手法が提案されてきている。移動体端末が十分なディスクを持たないことより、システム全体で同期的にチェックポイントを取得することは困難である。本論文では、移動体環境においてのチェックポイント取得方法として、固定端末は同期的に、移動体端末は非同期にチェックポイントを取得する複合型チェックポイント手法を提案し、その評価について述べる。

1 Introduction

According to the advances of communication and computer technologies, kinds of mobile stations like *personal data assistants* (PDA) are widely available. Each person can do the computation by using the mobile system while moving. Based on the mobile communications, new computation paradigms like *nomadic* computing [9] are proposed.

The information system is composed of fixed stations and mobile stations. The mobile stations move from one location to another in the network. There is a *mobile support station* (MSS) in each cell of the network. The mobile stations communicate with the MSS in the cell through the wireless channel. The MSSs and the fixed stations are interconnected by the high-speed network. The fixed stations do not move, i.e. located at the fixed location in the network. The connections with the mobile stations can be automatically maintained by the mobile protocols [11, 12, 13] even if the mobile stations are moving in the network. For example, applications in the mobile stations manipulate data in the fixed SQL servers. The applications are computed by the cooperation of the processes in the mobile and fixed stations. The communication links with the mobile stations are often disconnected since the mobile stations do not have so much capacity of the battery that they can communicate with other stations for a longer time and the wireless links are not so reliable as the cable links. The application processes have to be computed even if the connection is disconnected. In the *disconnected operations*, papers [3, 7, 8] discuss how to cache the data in the fixed station to the mobile one and how to keep the mutual consistency of data between the fixed station and the mobile one.

In order to realize the reliable distributed computation, the processes have to take the checkpoints where the local states in the processes are stored in the stable storage. The fixed stations can easily take the consistent checkpoints by using the synchronous distributed checkpointing protocols [6] since they have larger stable storages. On the other hand, the mobile stations may not take the checkpoints since they cannot have enough volume of stable storages or cannot access the stable storage due to the lack of the battery capacity. We assume that the MSS has enough volume of the stable storage to store the states of the mobile stations in the cell. The mobile stations take the checkpoints where the local states are stored in the MSS. The mobile station may fail to take the checkpoint due to the lack of the battery capacity and the movement to the outside of the cell. All the stations have to give up to take the checkpoints if some mobile station fails to take the checkpoint. In this paper, we propose a *hybrid checkpointing* where the mobile stations take asynchronously checkpoints while the fixed stations synchronously take the checkpoints. By the method, we can realize the fault-tolerant applications including the mobile stations.

In section 2, we present how difficult to take the synchronous checkpoint in the mobile computation environment. In section 3, we discuss the hybrid checkpoint and recovery method. In section 4, we present the evaluation.

2 Checkpointing

Suppose that multiple processes $p_1, \ldots, p_n$ are cooperated to do the distributed computation. In order to realize the fault-tolerant computation of $p_1, \ldots, p_n$, each p_i takes a local checkpoint c_i where the local state of p_i is stored in the stable storage named a *station* log l_i. If p_i is faulty, p_i is rolled back to c_i and then is restarted from c_i. A collection $\langle c_1, \ldots, c_n \rangle$ of the local checkpoints is a *global checkpoint* c, i.e. $c = \langle c_1, \ldots, c_n \rangle$. A message m is an *orphan* if m is received by a process but is not sent by any process. In Figure 1, p_i sends a message m after taking c_i and p_j receives m before taking c_j. m is an orphan at $\langle c_i, c_j \rangle$. c is *consistent* if there is no orphan message in c [6].

There are two ways to take the global checkpoint c among $p_1, \ldots, p_n$. One is a *synchronous* protocol[6] similar to the two-phase commitment (2PC) protocol [4]. Here, all the processes have to be suspended during the execution of the checkpoint protocol. The other is an *asynchronous* one, where the processes take the checkpoints independently of the other processes. However the domino effect [1] may occur. The paper [1] discusses a log-based method. Each p_i records in the log the messages sent and received by p_i after taking c_i. After rolling back p_i to c_i, p_i is recomputed from a point consistent with the others by using the messages stored in the log.

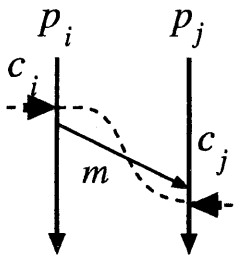

Figure 1: Orphan message

There are two kinds of stations, i.e. *fixed* station $F_1, \ldots, F_l$ and *mobile* ones $M_1, \ldots, M_l$ in the system. The fixed station is connected at the fixed location of the network. The mobile station is moving from one location to another. The network is composed of cells where there is one *mobile support station* (MSS). The mobile stations in a cell communicate with the MSS by using the wireless channel. The MSS forwards messages sent by the mobile station to the destinations and delivers the mobile stations messages destined to the mobile stations. The mobile station has the same address as long as in the same cell. The communication with the mobile station are automatically maintained by the cooperation of the MSSs even if the mobile station moves to another cell. The MSSs and the fixed stations are interconnected by the high-speed network.

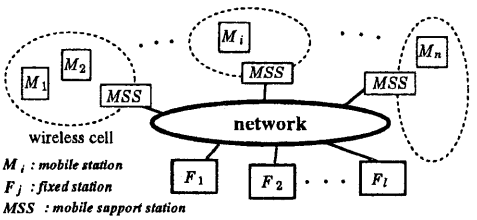

Figure 2: System model

Each the mobile station M_i does not have so much capacity of battery that M_i can communicate with the MSS for a longer time. Hence, M_i often disconnects the connection with other stations in order to reduce the power consumption while the applications are being computed in the mobile stations. Furthermore, M_i does not support enough computation power and storage capacity like disks. Hence, it is not easy for M_i to take the checkpoint by itself. In this paper, we discuss a way that the mobile station stores the local state into the log in the MSS at the checkpoint and messages sent and received by the mobile station after the checkpoint are also logged in the MSS.

Suppose that there are n mobile stations $M_1, \ldots, M_n$ in one cell of radius $100m$ where each station moves in an arbitrary way at a walking speed $1.3[m/sec]$. The communication link between the mobile station and the MSS nay be disconnected. Here, we assume that each link between M_i and the MSS is disconnected once two hours to reduce the power consumption of the mobile station. We also assume that it takes one minute for each M, to take the checkpoint because M_i sends all the data in the main memory, i.e. 32M bytes, to the MSS by using wireless channel of 2.8k bps. The MSS stores the local state of M_i in the log. We assume hat M_i fails to take the checkpoint if M_i gets out of the cell or the communication link is disconnected. A probability f that M_i fails to take a checkpoint is computed to be 0.01 from the assumption. In order to take the synchronous checkpoint, every M_i is required to make a success to take a checkpoint. If at least one station fails to take the checkpoint, every other station has to give up to take the checkpoint. The probability that $M_1, \ldots, M_n$ fail to take the synchronous checkpoints is $1 - (1 - f)^n$. Thus, it is difficult for the mobile stations to take the synchronous checkpoints.

3 Hybrid Checkpoint and Restart

3.1 Hybrid checkpoint

The distributed computation is realized by the cooperation of multiple processes in the mobile stations and fixed ones. The stations can exchange messages by using the mobile communication protocol [12, 13]. That is, the station can communicates with the others without being conscious of the locations of the stations. In this paper, we assume that the state of every station is changed only on receipt of messages. The station is assumed not to communicate with the users. Here, suppose that the mobile stations $M_1, \ldots M_n$ and the fixed stations $F_1 \ldots, F_l$ are cooperated. Each mobile station M_i is in a cell of the MSS S_i.

As presented before, it is difficult to take the synchronous checkpoints among the mobile stations. In this paper, we propose a *hybrid* checkpoint, which has the following properties:

(1) the fixed stations take the synchronous checkpoints, and

(2) the mobile stations take the asynchronous checkpoints.

Each mobile station M_i stores the local state in the MSS S_i. In addition, the messages sent and received M_i are logged in S_i.

Each M_i can takes c_i. That is, the mobile stations take asynchronously the checkpoints, i.e. independently of the other stations. M_i sends the local state to S_i. M_i fails to take c_i if M_i goes out of the cell or the battery of the station is exhausted during taking c_i. Hence, M_i takes c_i where M_i surely could take c_i. For example, M_i does not move a checkpoint taking c_i. M_i may take c_i only if it has enough capacity of the battery to take c_i.

M_i sends and receives messages after taking c_i. M_i sends messages to the MSS S_i and then S_i forwards the messages to the destination stations. M_i receives messages sent by other stations from S_i. Hence, S_i keeps in record the messages sent and received by M_i in the message log.

After rolled back to c_i, M_i does the same computation as ones executed before the rollback by using the state c_i and messages stored in the logs.

3.2 Checkpointing protocol

In the hybrid checkpointing, the fixed stations $F_1, \ldots, F_l$ take the synchronous checkpoints $cF_1, \ldots, cF_l$ while the mobile stations $M_1, \ldots, M_n$ take the asynchronous checkpoints $c_1, \ldots, c_n$, respectively. Each M_i communicates with the MSS S_i in the cell. For each M_i, there exists an agent process A_i in S_i. A_i stores the messages sent and received by M_i in the message log ml_i and takes the checkpoint c_i of M_i in the stable storage l_i if M_i requires A_i to take c_i. A_i also takes the checkpoint cA_i if the other fixed stations take the checkpoints in the synchronous way.

Suppose that M_i is first in a cell of the MSS S_{i1}. M_i moves from S_{i1} to S_{i2}. Thus, M_i moves from S_{ij} to $S_{i,j+1}$. M_i is currently in S_{ic_i} [Figure 3]. Here, S_{ic_i} is named a *current* MSS of M_i. Each S_{ij} has an agent A_{ij} of M_i. Each A_{ij} has a message log ml_{ij} where messages sent and received by M_i in the cell of M_{ij} are stored. The sequence of the logs $A_{i1}, \ldots, A_{ic_i}$ shows the sequence of messages which M_i has sent and received. Here, let $\{A_i\}$ denote the agent sequence $\langle A_{i1}, \ldots, A_{ic_i} \rangle$. We assume that M_i takes a checkpoint c_i where M_i is in S_{i1} and M_i has not taken any checkpoint in $S_{ij} (j \neq 1)$. Here, A_{i1} is named a *checkpoint* agent. $F_1, \ldots, F_l$ and the agents $\{A_1\}, \ldots, \{A_n\}$ of $M_1, \ldots M_n$ take synchronously checkpoints when each F_k takes cF_k and A_{ij} takes cA_{ij}

First, we discuss how each mobile station M_i takes a checkpoint c_i.

[Checkpointing in mobile stations]

(1) M_i sends a checkpoint request *Creq* to the current agent process A_{ic_i} in S_{ic_i}.

(2) On receipt of *Creq* from M_i, A_{ic_i} allocates the temporary log tl_i. Then, A_{ic_i} sends *Creq* to $A_{i1}, \ldots, A_{i,c_i-1}$ and the message log ml_{ic_i} to A_{i1}. On recept of *Creq* from A_{ic_i}, A_{ij} sends ml_{ij} to the checkpoint agent A_{i1}. A_{ij} removes ml_{ij}. On recept of all the message logs, A_{i1} stores the logs in the stable log l_{i1} and then sends back the reply *Crep* to A_{ic_i}. On recept of *Crep* from A_{i1}, A_{ic_i} sends to M_i.

(3) M_i transfers the local state to A_{ic_i} on receipt of *Crep*.

(4) On receipt of the state from M_i, A_{ic_i} stores the state with the location information of A_{i1} in the log tl_i □

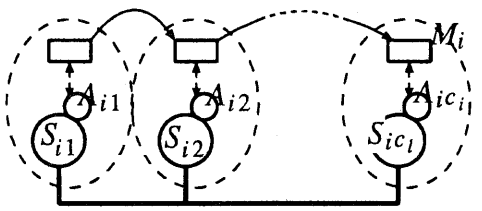

Figure 3: Mobile agents

There are two cases with respect to when M_i takes the local checkpoint c_i. That is, M_i takes the temporary log tl_i before A_{ic_i} takes the checkpoint cA_i or after cA_i. If tl_i is taken before cA_i, A_{ic_i} changes tl_i to the permanent checkpoint c_i. Then, A_{ic_i} starts to record messages sent and received by M_i in the message log ml_{ic_1}. Here, if the checkpoint agent A_{i1} is rolled back to the checkpoint cA_{i1}, M_i is restored by the state stored in A_{i1}. Then, M_{i1} does the computation and obtains the state consistent with A_{ic_i} by using the message logs $ml_{c1}, \ldots, ml_{ic_i}$. If c_i gets permanent, the

checkpoints preceding c_i are removed.

On the other hand, if tl_i is taken after cA_i, A_{ic_i} does not change tl_i to the permanent checkpoint c_i. If tl_i is changed to c_i, M_i cannot be rolled back to the stable consistent with A_{ic_i}.

We discuss how to take synchronous checkpoints among $F_1, \ldots, F_l$ and $\{A_1\}, \ldots, \{A_n\}$. Each F_k and A_{ij} take checkpoints cF_k and cA_{ij}, respectively, by using the two-phase commitment (2PC) protocol. Here, let $\{cA_i\}$ denote a set $\{cAc_1, \ldots, cA_{ic_i}\}$ of the checkpoints. $\langle cF_1, \ldots, cF_l, \{cA_1\}, \ldots, \{cA_n\}\rangle$ is consistent. Suppose that $\{A_j\}$ takes $\{cA_j\}$. There are the following cases.

(1) There is neither a checkpoint c_i nor a temporary checkpoint tc_i of M_i in $\{A_i\}$.
(2) There is c_i but is not tc_i in $\{A_i\}$.
(3) There is tc_i but is not c_i in $\{A_i\}$.

In (1), M_1 cannot be rolled back even if the fixed station and agents are rolled back.

The checkpoints most recently taken by all the stations are sure to satisfy the following properties:

(1) A checkpoint c_i of every mobile station M_i is taken in some MSS, i.e. A_{i1} and
(2) c_i is taken before the checkpoint of A_{i1}.

If some station is faulty, M_i is rolled back to the checkpoint c_i. Suppose that M_i takes a checkpoint c_i in A_{c1} and is in A_{ic_i}.

(1) M_i obtains the state stored at c_i from A_{i1}.
(2) M_i obtains a sequence of messages sent and received by M_i from the message logs $ml_{i1}, \ldots, ml_{ic_i}$ in $A_{i1}, \ldots, A_{ic_i}$, respectively.
(3) M_i restarts the computation from c_i.

3.3 Rollback protocol

We discuss how to rollback in the fixed stations and mobile stations if some station is faulty. The fixed stations and the agents are rolled back to the checkpoints most recently taken by the synchronous checkpointing protocol.

The mobile stations also have to be rolled back to the states consistent with $F_1, \ldots, F_l$ if some stations are faulty. The mobile station M_i takes asynchronously the checkpoint c_i. Hence, even if the mobile stations are rolled back to the checkpoints most recently taken, the checkpoints among the mobile stations neither denote the consistent states nor are consistent with the checkpoints of the fixed stations. In our method, the messages sent and received by M_i are recorded in the message logs $ml_{i1}, \ldots ml_{ic_i}$ of the agents $A_{i1}, \ldots, A_{ic_i} i$. ml_{ij} is stored in the agent A_{ij} in the MSS $S_{ij} (j = 1, \ldots, c_i)$. ml_{ij} includes messages sent and received by M_i after M_i takes c_i. M_i is first restored by the local state stored at c_i in A_{i1}. Then, M_i changes the state of M_i by using the messages received which are stored in the message logs.

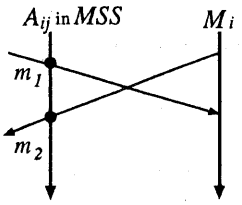

Figure 4: Message sequence

The agent A_{ij} receives messages sent to M_i in the same order as M_i and also forwards the destinations messages received from M_i in the same order as M_i. However, the sequence of the messages received and sent in A_{ij} may not be the same as M_i. For example, some station sends a message m_1 to M_i and M_i sends m_2 to the station. M_i sends m_2 before m_1 while A_{ij} receives m_1 before m_2. The message sequence in A_{ij} is different from M_{ij}. Hence, A_{ij} has to reorder the messages sent and received. Here, suppose that A_{ij} sends m_1 to M_i and receives m_2 from M_i. Suppose that A_{ij} sends m_2 to M_i before receiving m_2 from M_i. If m_1 causally precedes m_2 [5], the sequence of m_1 and m_2 is correct. However, if m_1 and m_2 are not causally ordered, A_i cannot decide in which order M_i sends m_2 and receives m_1. In this paper, we make the following assumptions.

[Assumption] Each mobile station M_i does not change the state on sending messages. □

That is, M_i changes the state only on receipt of messages. Here, if M_i receives the same messages in the same order, M_i obtains the same state as before the rollback.

Figure 5 shows two fixed station F_1 and F_2, and a mobile stations M_1 and M_2 supported by the MSS S. In Figure 5, the black triangles show the synchronous checkpoints taken by F_1, F_2, and S, and the black circles indicate the asynchronous checkpoints c_1 and c_2 taken by the mobile stations M_1 and M_2, respectively. M_1 and M_2 take c_1 and c_2 independently of the others stations. After taking c_1, M_1 sends a message m_1 to F_1 and m_2 to F_2. M_2 receive m, from F_2 after c_2. F_1 initiates the checkpoint procedure. F_1 sends a checkpoint request Cp to F_2 and S. On receipt of Cp, F_2 and S take the temporary checkpoints and then send Ack back to F_1. On receipt of Ack from F_2 and S, F_1 sends $Commit$ to F_2 and S. Here, F_1, F_2, and the S change the temporary checkpoint to the permanent ons.

Figure 6 shows fixed stations F_1, F_2, MSS S, and mobile station M_1. Suppose that F_1 is faulty. F_1 is rolled back to the checkpoint. F_1 sends the rollback request R_q to F_2 and S. F_2 and S are rolled back to the checkpoints. S sends the rollback request R_q with the state and the message log to M_1. M_1 is restored by the state and does the computation again by using the message log.

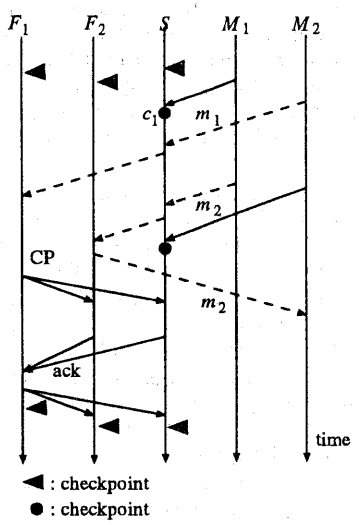

Figure 5: Checkpoint operation

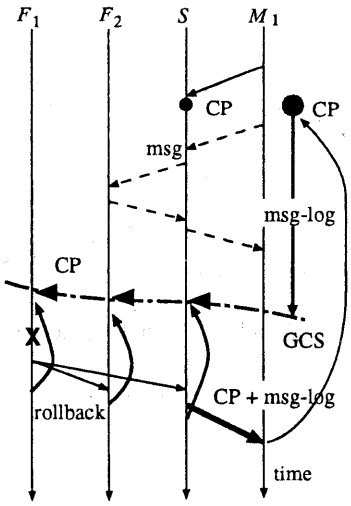

Figure 6: Rollback operation

The mobile station M_i has to find the checkpointed agent A_{i1} which has the checkpoint c_i and the agent A_{ij} which has the message log ml_{ij} in order for M_i to be rolled back. Each A_{ij} has a record on the preceding agent $A_{i,j-1}$ and the succeeding agent $c_{c,j-1}$.

4 Evaluation

We evaluate the hybrid checkpoints in terms of the total processing time of all the stations by comparing the following two ways for taking the checkpoints:

(1) Synchronous checkpoint : Every mobile station synchronously takes the checkpoint.

(2) Asynchronous checkpoint : Every mobile station asynchronously takes the checkpoint.

Here, it takes L seconds for each mobile station to take a checkpoint. We assume that no propagation delay between the stations. Suppose that $L = 300$ sec. Suppose that there are only n mobile stations $M_1, \ldots, M_n$.

In the synchronous checkpointing, the checkpoint is taken only if all the stations succeed to take the local checkpoint. If some mobile station fails, all the others have to throw away the effort to take the checkpoint and the stations restart the checkpoint procedure again. Let f be a probability that each mobile station fails to take the checkpoint, which is computed to be 0.01 in section 3. Each M_i takes 300 sec. to take the checkpoint by sending the state of M_i to the MSS. The probability that at least one mobile station fails during the checkpoint procedure is given $1-(1-f)^n$. The expected total time ET_S to take the checkpoint is $nL(1-f)^n(2+3(1-(1-f)^n)+4(1-(1-1)^n)^2+\ldots)/2 = nL(2-f)/[2(1-f)]$.

In the asynchronous checkpoint, each M_i asynchronously takes the checkpoint. If some M_i fails to take the checkpoint, M_i restarts the checkpoint procedure. Even if M_i fails, the other stations do not restart the checkpoint procedure. The expected time for each mobile station to take the checkpoint is $L(2+f+f^2)/[2(1-f)]$. Hence, the expected total time ET_A is $nL(2+f+f^2)/[2(1-f)]$.

Figure 7 shows ET_A and ET_S. ET_A is $O(n)$ while ET_S is $O(n^2)$. Figure 8 shows the ratio ET_A/ET_S for a number n of the mobile stations. Figure 9 indicates ET_S/ET_A for the probability f given $n = 1000$. These figures show that the hybrid checkpointing implies less overload than the synchronous one.

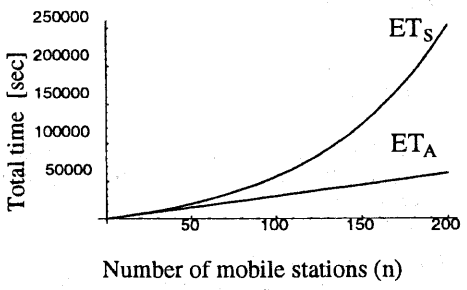

Figure 7: Expected total processing time

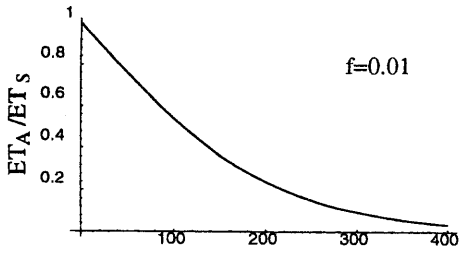

Figure 8: ET_A/ET_S

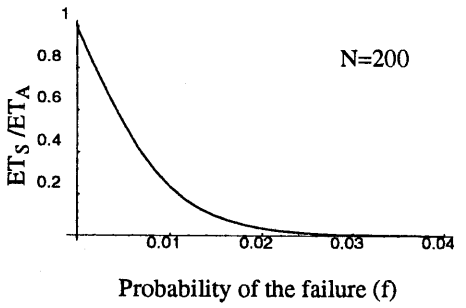

Figure 9: ET_S/ET_A

5 Concluding Remarks

We have discussed how to take the checkpoints and roll back the mobile stations and the fixed ones. We have newly proposed the *hybrid* checkpointing where the mobile stations asynchronously take the checkpoints and the fixed ones synchronously take the checkpoints. We have also shown that the hybrid checkpoint protocol discussed in this paper implies the less total processing time than synchronous one.

References

[1] Alvisi, L., Hoppe, B., and Marzullo, K., "Nonblocking and Orphan-Free Message Logging Protocols," *Proc. of IEEE FTCS*, 1993, pp. 145–154.

[2] Badrinath, B. R., Acharya, A., and Imielinski, T., "Structuring Distributed Algorithms for Mobile Hosts," *Proc. of IEEE ICDCS-14*, 1994, pp. 21–28.

[3] Barbara, D. and Imielinski, T., "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *Proc. of ACM SIGMOD*, 1994, pp. 1–12.

[4] Bernstein, P. A., Hadzilacos, V., Goodman, N., "Concurrency Control and Recovery in Database Systems," *Addison-Wesley*, 1987.

[5] Birman, P. Kenneth, and Renesse, V. Robbert, "Reliable Distributed Computing with the Isis Toolkit," *IEEE Comp. Society Press*, 1994.

[6] Chandy, K.M., and Lamport, L., "Distributed snapshots :Determining global states of distributed systems," *ACM Trans. on Computer Systems*, Vol.3, No.1, 1985, pp.63–75.

[7] Huang, Y., Sistla, P., and Wolfson, O., "Data Replication for Mobile Computers," *Proc. of ACM SIGMOD*, 1994, pp. 13–24.

[8] Kistler, J. J. and Satyanaranyanan, M., "Disconnected Operation in the Coda File System," *ACM Trans. on Database Systems*, Vol. 10, No. 1, 1992, pp. 2–25.

[9] Bagrodiu, R., Chu, W. W., Klienrock, L., and Popel, G., "Visim, Issues, and Architecture for Nomadic Computing," *IEEE Personal Cominication*, Vol. 2, No. 6, 1985.

[10] Prakash, R. and Singhal, M., "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 7, No. 10, 1996.

[11] Perkins, C., "IP Mobility Support," *Internet Draft:draft-ietf-mobileip-protocol-12.txt*, 1995.

[12] Tanaka, R. and Tsukamoto, M., "A CLNP-based Protocol for Mobile End Systems within an Area," *Proc. of IEEE ICNP-93*, 1993, pp. 64–71.

[13] Teraoka, F., Uehara, K., Sunahara, H., and Murai, J., "VIP: A Protocol Providing Host Mobility," *Comm. ACM*, Vol. 37, No. 8, 1994, pp. 67–75.