マルチメディア通信と分散処理 94－8
分散システム/インターネット運用技術 14－8
（1999．7．15）

Restart Protocol for Hybrid Checkpointing in Large-Scale Mobile Systems

Hiroaki Higaki, Katsuya Tanaka and Makoto Takizawa
Department of Computers and Systems Engineering
Tokyo Denki University

We have proposed a hybrid checkpoint protocol for stations to take their checkpoints and to be restarted in the mobile computing systems. Here, Two kinds of checkpoint protocols are combined. That is, fixed and mobile stations take their checkpoints synchronously and asynchronously, respectively. For supporting mobile stations to take checkpoints asynchronously and to be restarted consistently with the fixed stations, there are checkpoint agents in mobile support stations. In this paper, in order to support large-scale mobile computing systems, a communication-induced checkpoint protocol is adopted among the fixed stations. Especially, this paper discusses the restart protocol for supporting disjoint restart domain induced by the mobility of the stations. By using this protocol, it is realized for large-scale mobile systems to be fault-tolerant in the presence of stop faults in fixed and mobile stations.

大規模移動体システムにおける複合チェックポイント適用のためのリスタートプロトコル

桧垣 博章　田中 勝也　滝沢 誠
{hig, katsu, taki}@takilab.k.dendai.ac.jp
東京電機大学 理工学部 情報システム工学科

移動端末と固定端末が混在する移動体システムにおいて、高信頼なアプリケーション実行環境を構築する技術として、我々は複合チェックポイントを提案している。ここでは、リスタートに用いるチェックポイントが一意に決定可能であるという同期型チェックポイントの特性と、ネットワーク上の位置が時間とともに変化する、十分な資源を持たない、という移動端末の特徴に適合する非同期型チェックポイントの特性を考慮している。多数の端末からなる大規模移動体システムでは、同期型チェックポイントに要求される同期オーバヘッド(時間と同期メッセージの転送)が問題になる。そこで、固定端末では、communication-induced checkpoint protocol を適用する。ここでは、リスタートドメインという集合に含まれる端末が同時にリスタートすることが必要であるが、これに移動端末が含まれる場合には、ネットワーク上でリスタートドメインが連結ではなくなる場合がある。そこで、本論文では、移動端末のチェックポイント取得とリスタートを支援するチェックポイントエージェントの機能を拡張してこの問題を解決する。

1 Introduction

Information systems are getting distributed and larger by including various kinds of stations interconnected by local-area and wide-area networks, e.g. the Internet. Especially, many kinds of mobile stations like notebook computers, mobile computers, and personal data assistants (PDAs) are available. New computing paradigms like *nomadic computing* are also proposed according to the advance of the mobile stations. Hence, recent information systems are developed based on large-scale mobile computing systems including *fixed stations* and *mobile stations*. A fixed station is located at a fixed location in the network, i.e. it can always communicate with another station by using the same address. A mobile station moves from one location to another, i.e. its address may change during its computation. The network is divided into multiple *cells*. That is, a mobile station moves from one cell to another. There is a *mobile support station* (*MSS*) in each cell and a mobile station communicates with another station only through the MSS supporting the cell where the mobile station is supported.

In information systems, applications are realized by cooperation of multiple stations. Usually, these stations and the network interconnecting the stations are developed by using widely available products including engineering workstations, personal computers, mobile computers, Ethernets, routers, switching hubs, and so on. Mission critical applications cannot always be realized by using such products. Hence, it is important to discuss the way to make and keep the system so reliable and available. Checkpoint-restart is one of the well-known method to realize a fault-tolerant distributed systems. Each station sometimes takes its checkpoint during executing an application by storing its state information into its stable storage. If some station fails, the stations in the system are restarted from the checkpoints. To keep the system consistent even after the restart, a

set of checkpoints are required to be consistent. In order to take checkpoints consistently, there have been proposed two kinds of protocols: *synchronous* checkpoint protocols and *asynchronous* ones. Most checkpoint protocols have been designed to support only fixed stations. However, a mobile station has different properties from a fixed one, e.g. the mobility and the limitation of battery and storage. In order to realize reliable mobile information systems, a *hybrid* checkpoint protocol has been proposed [5]. Here, fixed and mobile stations take their checkpoints by using synchronous and asynchronous checkpoint protocols, respectively. However, all the fixed stations are required to take their checkpoints synchronously in the protocol. Hence, stations which do not need to be restarted are also restarted and exchanged synchronous messages are increased as the stations are increased. Communication-induced checkpoint protocols have been proposed to support large-scale distributed systems only including fixed stations [8]. In this paper, we propose a novel protocol realized by the combination of the communication-induced checkpoint protocol [4] and the hybrid checkpoint protocol in order to achieve fault-tolerance in large-scale mobile computing systems. Especially, we discuss a restart protocol achieved by the cooperation of fixed stations and *checkpoint agents* in MSSs supporting mobile stations.

The rest of this paper is organized as follows: In section 2, a system model of a large-scale mobile computing system is described. In section 3, the conventional protocols are discussed. In section 4, a novel protocol for supporting large-scale mobile computing systems is proposed.

2 System Model

A mobile computing system $\mathcal{S} = \langle V, L \rangle$ is composed of a set of stations $V = \{s_1, \ldots, s_n\}$ and channels $L \subseteq V^2$. An application is realized by cooperation of multiple stations communicating with each other by exchanging messages through the channels. We assume that each channel in L is reliable and bidirectional like TCP (Transmission Control Protocol). Communication events, i.e. a message-sending event $s(m)$ and a message-receipt one $r(m)$ of a message m, and local events occur in $\mathcal{S}$. A state of a station s_i is assumed to be changed only when a communication event occurs. A local state of s_i is determined by the initial state and the sequence of communication events occurring in s_i.

There are three kinds of stations: *fixed stations* F_i $(i = 1, \ldots, n(F))$, *mobile stations* M_l $(l = 1, \ldots, n(M))$, and *mobile support stations* (MSSs) S_s $(s = 1, \ldots, n(S))$. Each F_i is connected at a fixed location in the network. M_l moves from one location to another. If M_l is in a cell supported by an MSS S_s, M_l communicates with another station only through S_s by using a wired or wireless channel. S_s forwards messages from M_l to destination stations and delivers messages from the other stations to M_l. M_l has the same address as long as in the same cell. The connection with M_l is automatically maintained

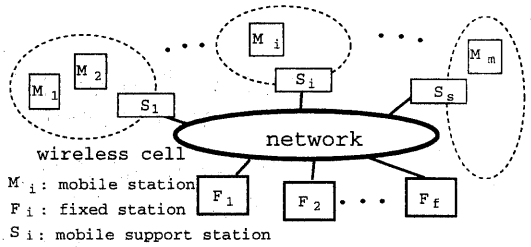

Figure 1: Mobile computing system.

by the cooperation of the MSSs even if M_l moves among the cells [10,12,13]. The fixed stations and the MSSs are assumed to be interconnected by a high-speed reliable network. M_l does not have so much capacity of battery that M_l can continue to communicate with an MSS S_s for a long period. Hence, M_l often disconnects the connection with another station in order to reduce the power consumption while the applications are being computed in M_l. Furthermore, M_l does not support enough computation power and storage capacity like disks storages. Hence, it is not easy for M_l to take a checkpoint by itself.

3 Hybrid Checkpoint Protocol

3.1 Consistent checkpoint

We overview the conventional checkpoint-recovery protocols. Each station $s_i \in V$ takes a *local checkpoint* c_i by storing the state information of s_i in the stable storage. A *global checkpoint* C is a set of local checkpoints taken by the stations in V, i.e. $C = \{c_1, \ldots, c_n\}$. If the stations take the local checkpoints and restart independently of the other stations, there may exist two kinds of inconsistent messages: *lost messages* and *orphan messages*. Here, suppose that s_i sends a message m to s_j. A message m is *lost* iff $s(m)$ occurs before c_i and $r(m)$ occurs after c_j. m is an *orphan* iff $s(m)$ occurs after c_i and $r(m)$ occurs before c_j. C is defined to be *consistent* iff there is neither lost nor orphan message [3]. If there exist orphan messages, $\mathcal{S}$ cannot be restarted consistently. However, if s_j stores m in a message log, s_j can take m from the log when s_j is restarted from c_j, that is, no message is lost and $\mathcal{S}$ is consistently restarted. Hence, C can be defined to be a *consistent global checkpoint* iff there is no orphan message.

Two kinds of protocols for taking consistent global checkpoints in $\mathcal{S}$ have been proposed: *asynchronous* and *synchronous* checkpoint protocols. In the asynchronous checkpoint protocols [1,7,15], each station in $\mathcal{S}$ takes local checkpoints independently of the other stations. If some station fails, the stations cooperate to find a consistent global checkpoint. On the other hand, in the synchronous checkpoint protocols [3, 9, 14], multiple stations are coordinated to take a consistent global checkpoint. The asynchronous checkpoint protocol implies less communication overhead for

taking checkpoints than the synchronous one because no communication is required among the stations. However, it takes longer time for the stations to be restarted in the asynchronous one because the stations have to exchange messages carrying the information of the local checkpoints. Moreover, if no consistent global checkpoint can be found, the stations have to be restarted from the initial state. It is called the *domino effect* [11]. In the synchronous checkpoint protocols, the stations can always be restarted from the most recent consistent global checkpoint and no domino effect occurs. The communication overhead for taking the checkpoint can be acceptable in the high-speed networks.

3.2 Hybrid checkpoint protocol

The computation in $\mathcal{S}$ is realized by cooperation of the fixed stations F_i $(i = 1, \ldots, n(F))$ and the mobile stations M_l $(l = 1, \ldots, n(M))$. The mobile stations are supported by MSSs S_s $(s = 1, \ldots, n(S))$. Each M_l is in a cell of some MSS S_s. Here, M_l is *supported by* S_s and S_s is the *current* MSS of M_l. The stations exchange messages by using a mobile communication protocol [10, 12, 13]. Each station can communicate 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 if a communication event occurs.

The synchronous checkpoint protocols have an advantage that the stations can be restarted without domino effect. However, it is difficult for multiple mobile stations to take local checkpoints synchronously [5]. Thus, a *hybrid checkpoint protocol* [5] has been proposed.

[Hybrid checkpoint]

- Fixed stations take their local checkpoints by using a synchronous checkpoint protocol. A collection of the checkpoints taken by the fixed stations is referred to as a *coordinated checkpoint.*
- The mobile stations take their local checkpoints by using an asynchronous checkpoint protocol. □

At a local checkpoint c_l^M of M_l, the state information of M_l is stored in the stable storage of the current MSS S_s. In addition, the messages sent and received by M_l are also stored in the stable storage of S_s. M_l fails to take c_l^M if the channel between M_l and S_s is disconnected. Thus, M_l can take c_l^M only if M_l does not move out of the cell and has enough capacity of the battery to take c_l^M. Therefore, M_l asynchronously takes the local checkpoint, i.e. independently of the other stations.

If some station fails, all the fixed stations are restarted from the coordinated checkpoint C. Each mobile station M_l is also restarted from its checkpoint c_l^M by restoring the state information stored in an MSS S_s. However, c_l^M might not be consistent with C. In a hybrid checkpoint protocol, S_s stores the messages exchanged between M_l and other stations after taking c_l^M. In a restart protocol, M_l recomputes the messages until M_l get a state consistent with C.

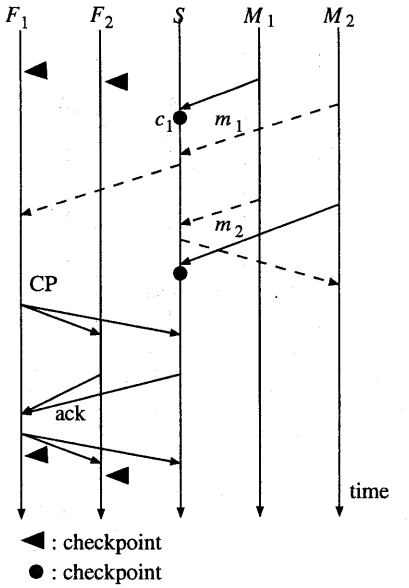

Figure 2: Checkpoint in Hybrid Protocol.

4 Protocols

4.1 Communication-induced protocol

In the conventional hybrid checkpoint protocol, all the fixed stations take their checkpoints simultaneously according to the protocol based on two phase commitment [9] for taking a consistent coordinated checkpoint. In a large-scale distributed system, i.e. there are many fixed stations in the network, high communication overhead for synchronization among the fixed stations is required. In order to restart the stations consistently, it is not always required for all the stations to be restarted from their checkpoints. Communication-induced checkpoint protocols [4, 8] have been proposed.

First, we define a *semi-consistent* coordinated checkpoint in $\mathcal{S}$.

[Semi-consistent] let G be a subset of a fixed stations $\{F_1, \ldots, F_{n(F)}\}$. A coordinated checkpoint $C(G) = \{c_i^F | F_i \in G\}$ is *semi-consistent* for G iff there is no orphan message for every channel of $F_i \in G$. □

Here, suppose that a fixed station F_i has taken a checkpoint c_i^F. If F_i sends a message m to another fixed station F_j after taking c_i^F, m is referred to as a *checkpoint message* of c_i^F to F_j. m carries an information that F_i has taken c_i^F. In order to take a semi-consistent coordinated checkpoint among a subset of the fixed stations $\{F_1, \ldots, F_{n(F)}\}$, each F_i takes its checkpoint c_i^F according to the following checkpoint rule:

[Checkpoint rule] F_j takes a checkpoint c_j^F just

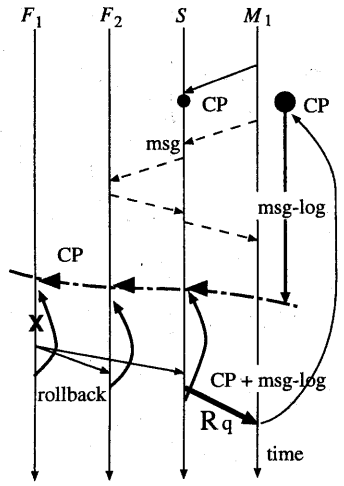

Figure 3: Restart in Hybrid Protocol.

before a message-receipt event $r(m)$ of a checkpoint message m in F_j. □

If F_i fails, it is not sufficient to restart only F_i from its checkpoint c_i^F because there may be orphan messages. In order to restart the system from a semi-consistent coordinated checkpoint, if F_i is restarted from c_i^F, an event e_j in F_j where $c_i^F \rightarrow e_j$ has to be canceled by restarting F_j. On the other hand, if no event e_j where $c_i^F \rightarrow e_j$ occurs in F_j, F_j is not required to be restarted from c_j^F.

[Theorem] According to the [Checkpoint rule], a semi-consistent coordinated checkpoint $C(G)$ is taken where the system S is kept consistent if only and all the fixed stations in G is restarted. □

In order to determine a set of fixed stations to be restarted if some fixed station F_i fails, we define a precedence relation among checkpoints and a restart domain as follows:

[Checkpoint precedence] Let c_i^F and c_j^F be checkpoints taken by F_i and F_j, respectively. Let e_i and e_j be events such that

- $c_i^F \rightarrow e_i$,
- $c_j^F \rightarrow e_j$, and
- there is no event e where $c_i^F \rightarrow e \rightarrow e_i$ or $c_j^F \rightarrow e \rightarrow e_j$.

Here, c_i^F *precedes* c_j^F ($c_i^F \Rightarrow c_j^F$) iff $e_i \rightarrow e_j$ [1]. □

[Restart domain] A *restart domain* $D(F_i)$ of F_i is a following set of fixed stations:

- $F_i \in D(F_i)$ if c_i^F has been taken by F_i. Otherwise, $D(F_i) = \emptyset$.
- $F_j \in D(F_i)$ if c_j^F has been taken by p_j and $c_j^F \Rightarrow c_k^F$ or $c_k^F \Rightarrow c_j^F$ where $c_k^F \in D(F_i)$. □

If all the stations in V are fixed stations, all the stations in each restart domain are connected. Hence, a message diffusion protocol can be used for sending restart request messages to all the stations in a restart domain.

[1] $\rightarrow$ means a causal relation among events [2].

4.2 Checkpoint information forwarding

In a mobile system $\mathcal{S} = \langle V, L \rangle$ where V includes both fixed and mobile stations, a restart domain $D(F_i)$ for a fixed station F_i is not always connected. This is because a mobile station M_l moves between cells after receiving a checkpoint message from F_i. For example, suppose that a mobile system $\mathcal{S}$ includes two fixed stations F_i and F_j, one mobile station M_l and two MSS S_s and S_t where there are two channels $\langle F_i, S_s \rangle$ and $\langle F_j, S_t \rangle$. Initially, M_l is supported by S_s. After taking a checkpoint c_i^F, F_i sends a checkpoint message m_i to M_l through S_s. Then, M_l moves to another cell supported by S_t and sends another checkpoint message m_j to F_j. On receipt of m_j, F_j takes a checkpoint c_j^F according to the checkpoint rule. Now, F_i and F_j are included in a restart domain which is not connected. Hence, a message diffusion protocol cannot be used to distribute restart request messages if one of the stations in this restart domain fails.

In order to solve this problem, *checkpoint agents* are required to cooperate. In this example, a checkpoint agent A_s^l of M_l in S_s holds the following information:

- A_s^l receives a checkpoint message from F_i.
- M_l moves to a cell supported by S_t.

On the other hand, another checkpoint agent A_t^l of M_l in S_t holds the following information:

- M_l moves from a cell supported by S_s.
- A_t^l sends a checkpoint message to F_j.

If F_i fails and is recovered, F_i sends a restart request message *Rreq* to A_s^l by using a message diffusion protocol. A_s^l forwards (or tunnels) *Rreq* to A_t^l according the above information. Then, A_t^l sends *Rreq* to F_j also by using a message diffusion protocol. By using this protocol, the restart request messages can be transmitted even in a disconnected restart domain.

For implementing this protocol, we extend an agent cooperation protocol for exchanging address information [6]. Here, suppose that a mobile station M_l moves from a cell supported by S_s to another one supported by S_t. Consider the case that M_l takes its checkpoint c_l^M while M_l is supported by S_s. The state information and the messages exchanged between M_l and other stations after taking c_l^M is stored in a stable storage in S_s. The messages exchanged between M_l and other stations while M_l is supported by S_t are stored in a stable storage in S_t. If M_l receives a restart request message, M_l has to collect the state information at c_l^M and the messages stored in S_s and S_t. Here, S_s and S_t exchanges their addresses, e.g. IP addresses and port numbers in TCP/IP network, through a home agent of M_l. In order to forward a restart request message, S_s informs S_t whether M_l has received a checkpoint message with the address information.

4.3 Hybrid checkpoint protocol for large-scale mobile systems

In a novel hybrid checkpoint protocol, a set G of fixed stations where $G \subset \{F_1, \ldots, F_{n(F)}\}$ take a semi-consistent coordinated checkpoint $C(G) = \{c_i^F | F_i \in G\}$ by using the communication-induced checkpoint protocol discussed in the previous subsection while the mobile stations $M_1, \ldots, M_{n(M)}$ take checkpoints $c_1^M, \ldots, c_{n(M)}^M$ by the asynchronous one.

Now, we discuss how each M_l takes c_l^M. Here, suppose that M_l is supported by S_s. The agent A_s^l in S_s takes a *tentative* checkpoint tc_l^M independently of the other stations. The state information required for M_l to be restarted from tc_l^M is carried by a tentative checkpoint request message *TCreq*. On receipt of *TCreq*, A_s^l stores the state information of M_l to the *tentative* state log tsl_l^M in the volatile storage of S_s.

[Tentative checkpoint tc_l^M in A_s^l]

1) M_l sends *TCreq* to A_s^l. *TCreq* carries the state information of M_l.
2) On receipt of *TCreq*, A_s^l takes tc_l^M of M_l by storing the state information to tsl_l^M.
3) If some agent A_t^l had taken another tentative checkpoint $tc_l'^M$ of M_l, A_s^l requires A_t^l to discard $tc_l'^M$.

A set $G \subset \{F_1, \ldots, F_{n(F)}\}$ of fixed stations take a semi-consistent coordinated checkpoint $C(G) = \{c_i^F | F_i \in G\}$ according to the following checkpoint rule:

- If F_i decides to take a checkpoint by such a trigger as a user request or a timeout and F_i has not yet taken a checkpoint, F_i takes c_i^F.
- If a message-receipt event $r(m)$ occurs in F_i where F_i has not yet taken a checkpoint and F_i receives a checkpoint message m from F_j, F_i takes c_i^F just before $r(m)$.

The former means that the checkpoint protocol can be initiated by multiple stations. By taking a checkpoint according to the latter, there is no orphan message in a channel $\langle F_i, F_j \rangle$.

Let $\langle A^l \rangle$ be a sequence of agents $\langle A_{s1}^l, \ldots, A_{sn}^l \rangle$ supporting M_l where A_{s1}^l has tc_l^M and A_{sn}^l is the current agent of M_l in the current MSS S_{sn}^l. If S_{sn}^l receives a checkpoint message from some fixed station, A_{s1}^l changes tc_l^M to a permanent checkpoint c_l^M by storing the state information in tsl_l^M to the stable state log sl_l^M. In addition, each $A_{s\alpha}^l$ $(1 \leq \alpha \leq n)$ stores the messages in the tentative message log tml_l^M to the stable one ml_l^M. The stable logs are stored in the stable storage while the tentative logs are stored in the volatile one.

[Permanent checkpoint c_l^M in $A_{s\alpha}^l$]

- If S_{sn}^l has not yet taken a permanent checkpoint and receives a checkpoint message, A_{sn}^l moves the messages from a tentative message log to a stable message log and sends a checkpoint request message *Creq* to $S_{s(n-1)}^l$.
- If $S_{s\alpha}^l$ where $1 < \alpha < n$ receives *Creq* from $S_{s(\alpha+1)}^l$, $A_{s\alpha}^l$ moves the messages from a tentative message log to a stable message log and forwards the checkpoint request message *Creq* to $S_{s(\alpha-1)}^l$.
- If A_{s1}^l receives *Creq*, A_{s1}^l moves the state information from tsl_l^M to sl_l^M.

In order to exchange address information and checkpoint information between $A_{s\alpha}^l$ and $A_{s(\alpha+1)}^l$, the following two protocols are applied:

[Disconnection from $A_{s\alpha}^l$]

- If M_l is disconnected from $A_{s\alpha}^l$, i.e. M_l moves out of a cell supported by $S_{s\alpha}^l$, M_l sends a message m_α to its home agent $\mathcal{A}^l$. m_α includes an address of $A_{s\alpha}^l$ and a variable C^l representing whether M_l has taken a permanent checkpoint.

[Connection to $A_{s(\alpha+1)}^l$]

1) If M_l is connected to $A_{s(\alpha+1)}^l$, i.e. M_l moves into a cell supported by $S_{s(\alpha+1)}^l$, M_l sends a message m_β including an address of $A_{s(\alpha+1)}^l$ to its home agent $\mathcal{A}^l$.
2) On receipt of m_β, $\mathcal{A}^l$ sends m_α and m_β to $A_{s(\alpha+1)}^l$ and $A_{s\alpha}^l$, respectively.
3) If C^l represents that M_l has taken a permanent checkpoint, messages that $A_{s(\alpha+1)}^l$ is going to send are checkpoint messages.

4.4 Restart protocol for large-scale mobile systems

We discuss how fixed and mobile stations are restarted. A fixed station F_i is restarted as follows:

[Restart in F_i]

- On receipt a restart request message *Rreq* from another fixed station or a checkpoint agent, F_i forwards the *Rreq* to all the neighbor fixed stations and checkpoint agents included in the same restart domain as F_i.

In order for $M_1, \ldots, M_{n(M)}$ to be restarted from the states consistent with a semi-consistent coordinated checkpoint C, checkpoint agents have to cooperate. Let $\langle A^l \rangle$ be a sequence $\langle A_{s1}^l, \ldots, A_{sn}^l \rangle$ of checkpoint agents for M_l. Suppose A_{s1}^l has a permanent checkpoint c_l^M, A_{sm}^l $(1 < m \leq n)$ has a tentative checkpoint tc_l^M, and A_{sn}^l is the current agent. That is, A_{s1}^l and A_{sm}^l receive *TCreq* from M_l and A_{so}^l $(1 \leq o \leq m)$ receives a checkpoint message. The messages exchanged between M_l and A_{sp}^l $(1 \leq p \leq o)$ are stored to ml_{sp}^l and used by M_l for obtaining a state consistent with C. A recovery protocol for M_l is as follows:

[Restart in M_l]

1) If S_{sn}^l receives *Rreq*, A_{sn}^l sends a state log request *SLreq* to A_{s1}^l and a message log request *MLreq* to every A_{sp}^l $(1 \leq p \leq o)$.

2) On receipt of *SLreq*, A^l_{s1} sends A^l_{sn} back a state log reply *SLrep* containing the state information at c^M_l in sl^M_l.
3) On receipt of *MLreq*, each A^l_{sp} $(1 \leq p \leq o)$ sends A^l_{sn} back a message log reply *MLrep* containing the messages in ml^l_{sp}.
4) If A^l_{sm} has tc^M_l, A_{sn} sends a tentative state log cancellation request *SLCreq* to A^l_{sm}.
5) On receipt of *SLCreq*, A^l_{sm} discards tc^M_l, i.e. discards the state information in tsl_{im}, and sends A^l_{sn} back a tentative state log cancellation reply *SLCrep*.
6) A^l_{sn} sends a message log cancellation request *MLCreq* to every A^l_{sq} $(m \leq q < n)$.
7) On receipt of *MLCreq*, A^l_{sq} discards the messages in tml_{iq} and sends A^l_{sn} back a message log cancellation reply *MLCrep*.
8) After receipt of *SLrep*, *MLreps*, *SLCrep* and *MLCreps* sent in steps 2), 3), 5) and 7), respectively, A^l_{sn} forwards these messages to M_l.
9) On receipt of the messages sent in step 8), M_l restarts from c^M_l by the state information carried by *SLrep* and recomputes the messages carried by *MLreps* to get the state consistent with C.

5 Concluding Remarks

In this paper, we propose a novel hybrid checkpoint protocol for supporting a large-scale distributed systems. Here, a communication-induced checkpoint protocol and an asynchronous checkpoint protocol are applied to fixed stations and mobile ones, respectively. A restart protocol for the hybrid checkpoint protocol is designed. For supporting a restart domain which is not connected, a cooperation among checkpoint agents by transmitting checkpoint information is designed. The cooperation is realized when a mobile station is disconnected and connected to the network, and when the system is restarted from the checkpoint.

References

[1] Bhargava, B. and Lian, S.R., "Independent Checkpointing and Concurrent Rollback for Recovery in Distributed Systems," Proc. of the 7th International Symposium on Reliable Distributed Systems, pp. 3–12 (1988).

[2] Birman, K. and Joseph, T., "Reliable communications in presence of failures," ACM Trans. on Computer Systems, vol 5, No. 1, pp. 47–76 (1987).

[3] . Chandy, K.M. and Lamport L., "Distributed Snapshots: Determining Global States of Distributed Systems," ACM Trans. on Computer Systems, Vol. 3, No. 1, pp. 63–75 (1985).

[4] Higaki, H., Sima, K., Tanaka, K., Tachikawa, T. and Takizawa, M., "Checkpoint and Rollback in Asynchronous Distributed Systems," The 16th Annual Joint Conference of the IEEE Computer and Communcations Societies (IEEE INFOCOM), pp. 1000–1007 (1997).

[5] Higaki, H. and Takizawa, M., "Checkpoint-Recovery Protocol for Reliable Mobile Systems," Proc. of the 17th IEEE International Symposium on Reliable Distributed Systems, pp. 93–99 (1998).

[6] Higaki, H., Shinchi, K., and Takizawa, M., "Recovery Protocol for Hybrid Checkpointing" IPSJ Technical Report, Vol. 98, No. 55, pp. 25–30 (1998).

[7] Juang, T.T.Y. and Venkatesan, S., "Efficient Algorithms for Crash Recovery in Distributed Systems," Proc. of the 10th Conference on Foundations of Software Technology and Theoretical Computer Science, pp. 349–361 (1990).

[8] Janssens, B. and Fuchs, W.K., "Experimental Evaluation of Multiprocesor cache-based error recovery," Proc. of the International Conference on Parallel Processing, pp. 505–508 (1991).

[9] Koo, R. and Toueg, S., "Checkpointing and Rollback-Recovery for Distributed Systems," IEEE Trans. on Software Engineering, Vol. SE–13, No. 1, pp. 23–31 (1987).

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

[11] Randell, B., "System Structure for Software Fault Tolerance," IEEE Trans. on Software Engineering, Vol. SE–1, No. 2, pp. 220–232 (1975).

[12] Tanaka, R. and Tsukamoto, M., "A CLNP-based Protocol for Mobile End Systems within an Area," Proc. of the International Conference on Network Protocols, pp. 64–71 (1993).

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

[14] Tong, Z., Kain, R.Y., and Tsai, W.T., "Rollback Recovery in Distributed Systems Using Loosely Synchronized Clocks," IEEE Trans. on Parallel and Distributed Systems, Vol. 3, No. 2, pp. 246–251 (1992).

[15] Venkatesh, K., Radhakrishnan, T., and Li, H.F., "Optimal Checkpointing and Local Recording for Domino-Free Rollback Recovery," Information Processing Letters, Vol. 25, pp. 295–303 (1987).