JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 29, 969-984 (2013)

DOI:10.1688/JISE.2013.29.5.10

Log Based Recovery with Low Overhead for Large Mobile Computing Systems*

PARMEET KAUR JAGGI[1] AND AWADHESH KUMAR SINGH[2]
[1]*Department of Computer Science*
Jaypee Institute of Information Technology
NOIDA, UP 201307, India
[2]*Department of Computer Engineering*
National Institute of Technology
Kurukshetra, Haryana, India

The article proposes a recovery protocol for applications in large mobile computing environment by combining movement based checkpointing with message logging. Since checkpointing is expensive, the focus of the scheme is to add only a low overhead to the normal application execution by reducing the number of checkpoints taken by a mobile host. For accomplishing this, the mobile system is divided into regions and a checkpoint is taken once when a mobile node enters a region and at most once while it moves within that region. The simulation results show that the scheme works well for large systems where the inter region movement of mobile hosts will be less as compared to intra region movement The recovery scheme is free from domino effect and aims to reduce the number of recovery related messages exchanged over the wireless media as well as the size of recovery related data stored with a mobile host. We show the correctness and effectiveness of our protocol in this presentation and demonstrate the simulation results.

Keywords: mobile hosts, checkpointing, message logging, recovery, region

1. INTRODUCTION

Distributed systems usually use checkpointing and message logging along with rollback recovery for providing fault tolerance. In the checkpoint-based recovery approach, the state of each process and the communication channel is stored onto stable storage *i.e.* checkpointed frequently during error free operation. At the time of failure, the system can be restored to a global and consistent checkpoint which will be a set of local checkpoints. There is no need to detect, log, or replay any non-deterministic events. On the other hand, the log based recovery schemes make it possible to continue the computation beyond the latest checkpoint. Such schemes consider the receipt of messages by any process and events internal to the process as non-deterministic events. The corresponding determinants of such events are logged into stable storage during normal execution. Checkpoints can also be taken to reduce the extent of rollback during recovery. At the time of recovery, the processes use the checkpoints and logged determinants to rerun the corresponding non-deterministic events to achieve a consistent state for the system.

A local checkpoint is a snapshot of a local state of a process. A local state of a process p_i is defined by the initial state of p_i and the sequence of events that have occurred at

Received May 12, 2011; revised September 26, 2011 & January 30, 2012; accepted March 18, 2012.
Communicated by Ce-Kuen Shieh.

* An earlier version of the work appeared in Proceedings of the 2nd International Conference on Advances in Communication, Networks and Computing, CNC 2011, Bangalore, India, March 10-11, 2011.

p_i. These events could be internal to p_i or the sending and receiving of messages from other processes by p_i. In a distributed system with N processes, a global checkpoint corresponds to a global state of the system and is formed by a set of local checkpoints, one from each process in the system. A global state is consistent if it denotes a state that could have been reached in the execution of the system. When considering the consistency of a global checkpoint G, we take into account a special category of messages called the orphan messages. These are the messages that have been recorded as delivered in G but not recorded as sent. A global checkpoint G is consistent only if there is no orphan message with respect to it *i.e.* there is none such message that is sent by a process after taking its local checkpoint but is received by another process before taking its local checkpoint. Thus an arbitrary set of local checkpoints at processes may not form a consistent global checkpoint. In order to solve the problem of capturing a consistent global state of a distributed system, Lamport's *happened before* relation [3], $\rightarrow$, is used to capture the causal relationships between events. Lamport's *happened before* relation [3] on events, $\rightarrow$, is defined for events a, b and c as;

- $a \rightarrow b$ if a and b are within the same process and a occurred before b,
- $a \rightarrow b$ if a is the send event of a message and b is the corresponding receive event of the same message by some other process,
- if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$.

Each checkpoint of a process is assigned a unique sequence number. Let $C_{i,k}$ ($0 \leq i \leq N - 1$, $k \geq 0$) denote the kth checkpoint at a process p_i. If $S(m)$ is the send and $R(m)$ is the receive event of a message m, then $S(m) \rightarrow C_{i,k}$ if m was sent by p_i before taking $C_{i,k}$ and $R(m) \rightarrow C_{j,l}$ if m was received and processed by a process p_j before taking $C_{j,l}$. A global checkpoint G is a set of checkpoints and may be defined as

$G = \{_{0 \leq i \leq N-1} \cup_i C_{i,k}\}$ and is consistent if it satisfies the following condition:

$$\forall m\text{: if } \exists i\text{: } R(m) \rightarrow C_{i,k} \text{ then } \exists j\text{: } S(m) \rightarrow C_{j,k}. \quad (1)$$

The rollback recovery protocols based on checkpointing and message logging for wired distributed systems are not directly applicable to mobile computing systems due to the differences in the environment in which these systems operate. In a mobile computing system the mobile hosts face the additional constraints of limited battery power and data storage. Their location is not fixed and it may happen that a mobile host recovers in a cell different from which it crashed in. The mobile devices may even lose connection with other systems voluntarily or involuntarily. [1] Due to the above reasons, fault tolerance schemes for the mobile computing systems should meet additional conditions as compared to the distributed systems with only fixed hosts. A mobile host (MH) needs to be located before a message can be delivered to it. The recovery procedure needs to be aware of the location of the mobile device to reduce the 'search cost' [2]. Some nodes may not be available at the time of recovery. The recovery of the application state should not be affected by the unavailability of some nodes. The data storage on the mobile hosts is limited, so usually the storage at Base Transceiver Station (BTS) in mobile computing systems is used for storing the checkpoints and message logs of MHs in its cell. However the bandwidth of the wireless link between the MH and the BTS is

relatively less than that of the fixed links in the static network. A recovery protocol that reduces the size of messages stored on the mobile hosts or decreases the number of messages passed over the wireless media will be efficient [2].

In our work, we employ movement based checkpointing and sender based message logging for the consistent recovery of a mobile host subsequent to a failure. The recovery protocol is free from domino effect as the global checkpoint is taken in a coordinated fashion. We define the parameters Checkpoint Expense and Control Expense to evaluate the overhead caused to normal application execution by our recovery scheme. A preliminary version of the work presented here has appeared in [4]. This paper is organized as follows. The system model is described in the next section. Related work done in the area of message logging is discussed in the subsequent section. Then we present the proposed message logging, checkpointing and recovery protocols for mobile computing systems. Storage management issues at the BTS are discussed. We evaluate our protocol to compare with other contemporary protocols and give its proof of correctness. Finally we conclude the presentation.

2. SYSTEM MODEL

We consider a typical mobile computing environment consisting of mobile hosts and a wired network of some fixed computers. The fixed host may be a regular host or a Base Transceiver Station (BTS), also referred to as the Mobile Support Station. The communication between the MHs and the wired network takes place through the base transceiver stations. A Base Transceiver Station manages the mobile hosts within its cell. A cell is defined as the geographical area covered by a BTS. To communicate with an MH that is not in the same cell, the source MH sends a message to its local BTS using wireless links. This BTS forwards the message over the fixed or wired network to the BTS of the destination MH. The receiving BTS sends the message to the destination MH over the wireless media.

It is assumed that each MH runs a single process and the processes are fail-stop. We use the terms node, host and process interchangeably. The processes communicate with each other only by exchanging messages. The underlying fixed network is FIFO, asynchronous and reliable with unpredictable but finite transmission delays. No message will be lost in the channel. The mobile hosts may move between cells, thus, necessitating a transfer of control from one mobile support station to another. A number of BTSs are

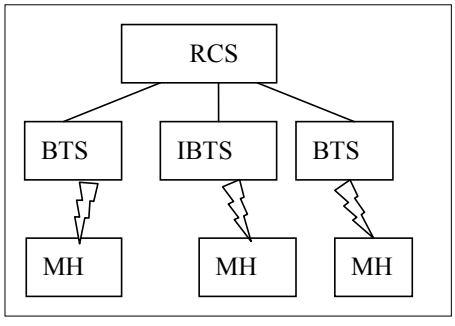

Fig. 1. A region.

controlled by a single switch, the Mobile Switching Centre (MSC).We define a region as the set of all BTSs controlled by a MSC as shown in Fig. 1. The concept of a region is used for the mobile environment such that a given fixed host keeps track of the mobile hosts present in its region. This reduces the cost to locate any mobile device as the region based hierarchical approach helps us to decrease the recovery related information being carried by the MH at any given time.

3. RELATED WORKS

Log-based rollback-recovery (also referred to as the message logging) protocols ensure that there are no orphan processes in the system after the recovery process. There will not remain any process whose state depends on a nondeterministic event that cannot be reproduced during recovery. Various protocols differ in the failure-free performance overhead, number of messages exchanged, the recovery process and garbage collection. There has been substantial amount of work done in the past to achieve an efficient scheme for recovery of applications in the mobile computing systems. Some of the work involves only checkpoints, while others employ message logging.

The work presented in [1] describes an uncoordinated checkpointing scheme for mobile hosts where multiple MHs can arrive at a global consistent checkpoint by using application messages and without extra coordination messages. An MH checkpoints local state whenever it moves to a new cell, prior to disconnecting from the network or when required by the protocol. The application messages are piggybacked with control information. The checkpointing algorithm can select a set of local checkpoints which form a consistent global checkpoint and obtain the location of these local checkpoints. However, this paper does not describe how failure recovery is achieved. A failure recovery scheme for mobile database applications based on movement-based checkpointing and logging is employed in [5]. A checkpoint is taken only after a threshold of mobility handoffs has been exceeded. The optimal threshold is determined dynamically. But this paper doesn't consider the case where a mobile host may not be moving much within cells. The protocol described by [6] uses time to indirectly coordinate the creation of new global state without the use of message exchanges. A hybrid checkpoint recovery scheme is proposed in [7]. The fixed hosts take a coordinated checkpoint while the mobile stations take local checkpoints asynchronously. A mobile host leaves an agent on every MSS it registers with. These agents help in achieving a consistent global state for the system.

A detailed study of pessimistic, optimistic and causal message logging protocols is provided in [8]. The consistency requirement of the system after recovery is expressed in the terms of orphan processes and an always-no-orphans condition is derived. A recovery scheme utilizing independent checkpointing and optimistic message logging for reliable as well as unreliable systems is proposed in [9]. A causal message logging protocol with independent checkpointing for mobile nodes is presented in [10]. It deals with the constraints of the mobile nodes and describes a low-cost failure-free mechanism for checkpointing. A mobile node only maintains the latest checkpoint of each process during its handoff process. The protocol uses two garbage collection schemes to remove log information of mobile nodes.

Three sources of overhead due to message logging have been identified in [11] Firstly, each message needs to be copied to the local memory of the process. Secondly,

the volatile log is regularly flushed to stable storage to free up space. Thirdly, message logging nearly doubles the communication bandwidth required to run the application for systems that implement stable storage via a highly available file system accessible through the network. The authors consider sender based logging more efficient than receiver-based logging because the copying of log to stable storage can take place after sending the message over the network. Additionally, it is suggested that by combining the message logging with the implementation of the communication protocol, the overhead of logging can be removed. The authors in [12] show that the sender based message logging technique has a very low overhead in providing fault tolerance in distributed systems. Further, it has been shown in [13] that the failure-free performance of coordinated checkpointing and message logging is better than of uncoordinated checkpointing and message logging. The work in [14] presents a communication-induced checkpointing protocol and an asynchronous recovery scheme. The recovery protocol uses selective message logging at the mobile support station to handle the messages lost due to rollback. Recently, fault tolerance schemes employing checkpointing and message logging in mobile computing systems have been studied and analyzed in [15, 16]. The authors have described the challenges due to the mobile environment and compared the application of coordinated, uncoordinated, communication induced and log based checkpointing protocols in such systems.

4. PROPOSED SCHEME

The scheme proposed in this paper employs sender based message logging along with movement based checkpointing to reduce the number of checkpoints taken by a mobile host. The mobility of a node is used for deciding when it needs to take a checkpoint. We utilize the storage at a Base Transceiver Station to store the checkpoints and message logs of the mobile hosts. Moreover sender based logging scheme avoids the extra copying of the message to the BTS. If the message logging is combined with the underlying communication protocol, then no extra overhead is placed for logging the messages at the sender BTS. The size of the information stored at the mobile host is kept at a low as compared to [5, 9] and few messages are exchanged via the wireless media. At most one checkpoint is required to be stored at a BTS per MH.

We define a region as the set of all BTSs controlled by a MSC.A fixed host is introduced as the Regional Coordinator System (RCS) for a region. This RCS may be the MSC or any other fixed host specifically designated for this purpose. An MH entering the cell of a BTS is registered with that BTS. However, the first BTS with which the mobile host registers when it enters a region is designated as the InitialBTS (IBTS). When a mobile host moves to a new region, the first BTS with which it registers becomes the new IBTS for the MH. The IBTS will be responsible for managing the message log of the MH till the MH is in the same region. In addition to the message log, the IBTS also maintains a list of processes from which the MH has received messages since the last checkpoint. This list is designated as RCVD_LIST.

The RCS maintains a list of the mobile hosts in the region at any given time by using a data structure, CURRENT_REGISTRATIONS, of the form <MH, current BTS of the MH>. This data structure has to be updated by a BTS when an MH registers with it. An MH identifies itself to any BTS with <own MH id, id of its initial BTS of the region,

id of its previous BTS>. The size of this data stored at the MH is less than storage required at the MH in protocols like [5, 9] where the entire log set, containing a list of all the previous BTSs after the last checkpoint, is stored at the MH. We use a log unification scheme such that the complete list of BTSs with which the MH has registered after the last checkpoint need not be stored with the MH. The message logging and unification scheme is described next.

4.1 Message Logging

(1) When an MH enters a network

Action taken by an MH

- MH registers with a BTS. As there is no previous BTS entry in its own data structure, the current BTS becomes its IBTS for the current region.
- The MH updates its own data structure to denote the current BTS as its IBTS.

Action taken by the IBTS

- IBTS updates the CURRENT_REGISTRATIONS data structure at the RCS to inform that the MH is currently registered with it.
- Further messages being sent by the MH are logged at the IBTS. No additional work needs to be done for this logging because the messages are sent through the BTS only. The senders of any messages destined to the MH are put in the corresponding RCVD_LIST.

(2) When there is a handoff from the IBTS to another BTS within the same region.

Action taken by the MH

- The MH updates its own data structure to save the id of the previous BTS, which is the IBTS in this case.

Action taken by the IBTS

- A tentative checkpoint is taken at the IBTS.

Action taken by the current BTS

- The current BTS updates the entry for the MH at the RCS to show that the MH is registered with it.
- Further messages sent by the MH are logged at its current BTS using sender based logging and the senders of the messages received by the BTS destined for the MH are saved in the corresponding RCVD_LIST.

(3) When there is a handoff from any BTS to another BTS within the same region. Considering the handoff pattern of an MH within a region like.

$$\text{IBTS} \rightarrow \text{BTS}_i \rightarrow \text{BTS}_j \rightarrow \text{BTS}_k$$

Say the MH has moved from BTS_j to BTS_k,

Action taken by the MH

- The MH updates its own data structure to show the id of the previous BTS. Here the MH will make BTS_j its previous BTS.

Action taken by the current BTS

- If the previous BTS as identified by the MH in the handoff pattern is the IBTS, then no control message is passed.

- But if the previous BTS as identified by the MH in the handoff pattern is not the IBTS, then the current BTS sends a *delete_log* message to the previous BTS. So in this case, BTS_k sends a delete log message to BTS_j.
- The current BTS updates the CURRENT_REGISTRATIONS at the RCS to inform the presence of the MH with it.

Action taken by the previous BTS

- On receiving the *delete_log* message, the previous BTS then appends its log and the RCVD_LIST for the MH at IBTS and deletes its own copy of the log and the RCVD_LIST. In this case, BTS_j appends its log for the MH to the IBTS and deletes its own copy of the log and the RCVD_LIST.

As a special case, we consider the scenario that the next BTS with which a mobile host is registering is the same as its previous BTS. In such a case, the present log and the RCVD_LIST are appended to previous' log and RCVD_LIST for the MH and then own copy is deleted. Thus the next BTS will not have an entry for the previous BTS. Thus, whatever be the movement pattern of an MH within a region, at a given instant only the message logs at the current and previous BTS are not available at the IBTS. The message logging ensures that the log at any other BTS is logged at the IBTS as the MH moves.

(4) When there is a handoff from a BTS in one region to a BTS outside the region.

The first BTS with which it registers becomes the initial BTS.

Action taken by the MH

- The MH updates its data structure to show that the id of the IBTS for the region. Previous BTS is set to null.

Action taken by the new IBTS

- A tentative checkpoint is taken at this BTS which then logs further messages being sent by the MH and maintains the RCVD_LIST for the MH.
- The IBTS stores the record of this MH at the RCS of this region.

Action taken in the previous region

- The previous IBTS deletes its tentative checkpoint for the MH.
- The RCS of the previous region deletes the record of this MH in its CURRENT_ REGISTRATIONS.

Hence, at any time at most one tentative checkpoint is maintained for an MH.

(5) On disconnection of an MH from the current BTS

If an MH wants to disconnect voluntarily, it sends a *disconnect* message to its current BTS. The BTS will save this information till either the MH reconnects again with the same BTS or a checkpoint request reaches the BTS. While the MH is disconnected, if a checkpoint request reaches the BTS, it participates in the message log unification for the MH When the MH reconnects; the BTS changes its status to connect.

4.2 Global Checkpoint

A global checkpoint is taken in a coordinated fashion and each such checkpoint is assigned a unique sequence number. The checkpointing can be initiated by any RCS as follows:

- An RCS sends a checkpoint request to all the other RCSs and fixed hosts $F_1 \ldots F_n$.

Actions taken on receipt of this request

- Fixed hosts take tentative local checkpoints and return a message to inform when they have taken their check points.
- Every RCS sends a message to its BTSs.
- The current BTS of the mobile host sends a message to previous BTS of the MH to append the log and the RCVD_LIST for this MH at the initial BTS of the MH.
- The present BTS also unifies its own message log and the RCVD_LIST at the initial BTS.
- A reply is then sent back to the initiator RCS.

Actions taken on receipt of reply from the hosts

- The RCS asks the hosts to finalize the checkpoints.
- The fixed hosts and the BTSs make their checkpoints permanent.

4.3 Storage Management at the IBTS

If many mobile hosts register with the same BTS for the first time in a region, that BTS may get overloaded and exhaust its storage. To avoid overloading of any BTS acting as the IBTS, the IBTS for any given MH can be reallocated. Any IBTS can send a *Relocate_MH* (*list of MHs*) message to the RCS of the region. The RCS finds the BTS with minimum number of entries in its CURRENT_REGISTRATIONS. For every MH included in the *Relocate_MH* message sent to the RCS, the log is unified at the previous IBTS by the approach described above. A new tentative local checkpoint is taken at the new IBTS by using the tentative checkpoint at the previous IBTS along with the unified message log. Then the previous checkpoint and message log at the previous IBTS are discarded. This is possible even if the MH is currently disconnected. The MH will change its ID to reflect the allocation of the new IBTS.

4.4 Handling Unreliable BTSs

In mobile computing systems, each BTS broadcasts its advertisement message via its wired or wireless network interface every few seconds. Therefore, a RCS can detect if any BTS has failed. The RCS takes over the failed BTS and restores the recovery related information about all the MHs earlier registered with the failed BTS from its stable storage. After that, the RCS informs the MHs registered with the failed BTS and other BTSs in the region that it has replaced the failed BTS. The RCS performs the role of the failed BTS till it recovers and resumes normal operation. Therefore, the protocol handles the BTSs' failures effectively assuming reliable RCS.

5. RECOVERY

The recovery related data of a process is distributed amongst its IBTS, last BTS and the previous to last BTS at the time of failure. This data includes the MH's latest checkpoint and a list of processes, the RCVD_LIST, from which it had received messages after its latest checkpoint. The recovery related data can be located as follows:

- If the MH recovers in the same region in which it crashed, then the RCS of the region has a record of the MH and its BTS before the crash in its region. The log at that BTS and its previous BTS are unified at the IBTS of the MH in the region.
- If the MH recovers in a region different from the region in which it crashed, then the RCS of the current region sends a *locate_MH* message to other RCSs. The RCS with the latest record of the MH informs the current RCS using a message over the fixed media for the recovery process.

Once the recovery related data is obtained, the MH rollbacks to the latest checkpoint. The current BTS sends a message to each process in the RCVD_LIST to resend the messages sent to this MH after the latest checkpoint. As a sender based logging scheme is used, the processes in the RCVD_LIST have a log of the messages sent to the MH at their BTSs. The message log of each sender process is unified at its corresponding IBTS and required messages are resent to the MH. The MH plays these messages and recovers to a consistent state. During recovery, it may resend out messages which may have been earlier received at their destinations. Such messages can be discarded at the destination based on message sequence numbers.

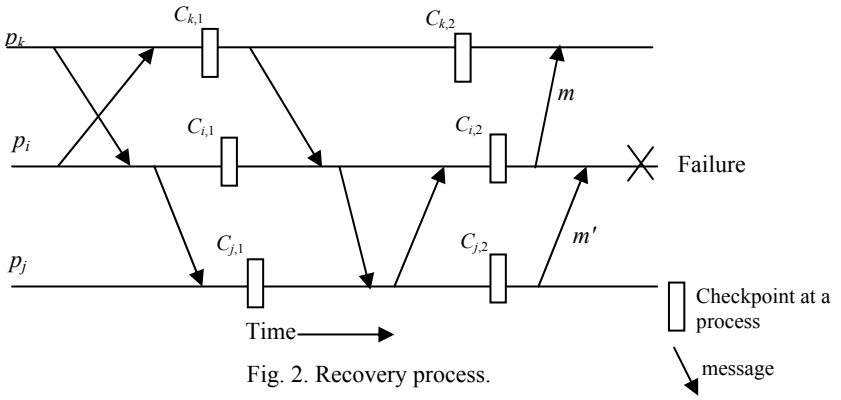

Fig. 2. Recovery process.

The recovery process at process p_i is shown in Fig. 2. Once the process p_i needs to recover after the failure, it obtains its latest checkpoint and RCVD_LIST from its IBTS prior to failure. Then p_i rollbacks to its latest checkpoint, here $C_{i,2}$, and replays the logged messages. Hence any process p_k that received messages from p_i before p_i's failure need not rollback. However, the message log of a process such as p_j, which is in p_i's RCVD_ LIST is unified at its corresponding IBTS. Then the messages such as m', sent by p_j after its latest checkpoint $C_{j,2}$ but un-received due to the roll back by p_i, are resent.

6. PERFORMANCE STUDY

6.1 Parameters for Evaluation

To evaluate the overhead caused to the application execution because of the recovery

protocol, we define two parameters as follows:

Definition 1: *Checkpoint Expense*(*Chk_exp*)
It is the increase in execution expense of a process due to a single checkpoint taken at a BTS.

Definition 2: *Control Expense*(*Control_exp*)
It is the increase in execution expense of a process due to the control messages exchanged during the checkpointing process.

6.1.1 Calculation of Chk_exp

A tentative checkpoint is taken at the IBTS when there is a handoff from the IBTS to another BTS in the same region or if the MH moves to a new region. The expense for taking a single tentative checkpoint = 1 * E_w; where E_w is the expense of sending a message on the wireless media from an MH to its BTS. The expense for unification of log at the IBTS by the any other BTS = 1 * E_f; where E_f is the cost of sending a message across a fixed network. Hence, for a single MH in one region, say R_x

$$Chk\_exp = E_w + (N_x - 1) * E_f. \quad (2)$$

Here N_x is the number of registrations for an MH in the region R_x (this can vary from one region to another).

Thus only one additional wireless message, due to recovery procedure, is passed only at the time of taking a tentative checkpoint at a BTS *i.e.* is $O(k)$ where k is the no of regions in which the MH moves.

If an MH moves in k regions, then

$$Total\_Chk\_exp = k*E_w + {}_{x=1}^{k}\Sigma(N_x - 1)*E_f. \quad (3)$$

As compared to the proposed scheme, if checkpoints are taken periodically, such as in [9], then the Total number of checkpoints = ${}_{x=1}^{k}\Sigma N_x/t$; where t is the time period between consecutive checkpoints.

$$Total\_Chk\_exp = ({}_{x=1}^{k}\Sigma N_x/t) * E_w \quad (4)$$

Further, if checkpoints are taken based on mobility rates as in [5], then Total number of checkpoints = ${}_{x=1}^{k}\Sigma N_x/M$; where M is a factor dependent on mobility

$$Total\_Chk\_exp = ({}_{x=1}^{k}\Sigma N_x/M) * E_w. \quad (5)$$

If checkpoints are taken each time an MH moves out of a cell as in [1], then Total number of checkpoints = ${}_{x=1}^{k}\Sigma N_x$

$$Total\_Chk\_exp = ({}_{x=1}^{k}\Sigma N_x) * E_w. \quad (6)$$

Since $k < {}_{x=1}{}^{k}\Sigma N_x$, the overhead due to the proposed scheme is lower than the other schemes discussed above.

6.1.2 Calculation of Control_exp

If N_x is the number of registrations for an MH in the region R_x, the following messages contribute to the *Control_exp* in the region R_x.

- The updating of current BTS of an MH at the RCS = N_x.
- *Delete_log* messages passed = $N_x - 3$.
- Control messages from the RCS to the BTSs in the region to take a global checkpoint = N_x in a region.
- The message passed from the current to previous BTS in a region for the unification of log at the IBTS at the time of global checkpoint = 1.
- The search by an RCS of the last RCS of an MH at the time of recovery = (number of *RCSs* − 1) * E_f.

If k is the no of regions in which the MH moves, then

$$Control\_exp = {}_{x=1}{}^{k}\Sigma(N_x * E_f + (N_x - 3) * E_f + N_x * E_f + E_f) + (\text{number of } RCSs - 1) * E_f.$$

Thus,

$$Control\_exp = {}_{x=1}{}^{k}\Sigma((3N_x - 2) * E_f) + (\text{number of } RCSs - 1) * E_f. \quad (7)$$

6.2 Proof of Correctness

The correctness of our protocol is guaranteed by the following two theorems.

Theorem 1: No orphan messages can be generated by the system.

Proof: For nondeterministic event, e, of the receipt of a message at the process p, define:
(1) Depend(e) is the set of processes affected by the nondeterministic event e. This set consists of p, and any process whose state depends on the event e according to Lamport's happened before relation.
(2) Log(e) is the set of processes that have logged a copy of e's determinant in their volatile memory.
(3) Stable(e), a predicate that is true if e's determinant is logged on stable storage.

A process p becomes an orphan when p itself does not fail and p's state depends on the execution of a nondeterministic event e whose determinant cannot be recovered from stable storage or from the volatile memory of a surviving process. Formally

$$\forall(e): \neg Stable(e) \Rightarrow Depend(e) \subseteq Log(e). \quad (8)$$

This property is called the *always-no-orphans* consistency condition [8].

In the proposed recovery scheme, if a BTS_j has logged the receipt of a message for an MH registered with it, then some BTS_i must have logged the message content and the corresponding send event in its stable storage. This is because sender based pessimistic logging is utilized along with the communication protocol itself. Hence the determinant of a nondeterministic event is always available upon recovery and there can not be any orphan messages. Thus our protocol satisfies the *always-no-orphans* condition.

Theorem 2: The recovery of a mobile host is consistent assuming reliable BTSs and is free from domino effect.

Proof: The message logging scheme ensures that the message log gets unified in the correct sequence, as per the handoff pattern for the MH, at the IBTS. Upon recovery, the latest checkpoint of the MH is available at the last IBTS of the MH. This IBTS may be in the current region of the MH or any other region. However, the IBTS can be located and then the message log is unified at that IBTS. This data is sent to the current location of the MH. To reconstruct the state as before failure, the processes from which the MH had received messages prior to failure are requested to resend the messages. The MH roll-backs to its latest checkpoint and replays those messages to recover to a consistent state. Thus recovery of the MH is consistent assuming reliable BTSs. Further no other process is required to rollback and thus domino effect of unbounded rollback propagation is not possible.

6.3 Comparison with Other Related Work

The protocol proposed above combines movement based checkpointing with sender based message logging for achieving independent recovery of a mobile host. To evaluate the performance of the proposed recovery protocol, we have compared our scheme with the scheme proposed in [5], the periodic checkpointing scheme for a system with reliable BTSs in [9] and the communication induced checkpointing and selective message logging in [14]. Table 1 summarizes the salient features of a checkpointing and rollback based recovery protocol across the three schemes.

6.4 Simulation Model and Results

We have used a mobile computing system with the mesh cell configuration, consisting of $N \times N$ [$N = \{4, 5, 6\}$] cells where all cells are of the same size [9]. Each cell has eight neighboring cells and all MHs in this cell are assumed to be registered with the same BTS. An MH may move from one cell to another neighboring cell. We select the next cell for the MH's movement out of the eight possible neighbors randomly. The time interval between two handoffs follows an exponential distribution with an average of $1/\lambda_h$ where $\lambda_h = 0.05$ as in [9]. We have simulated our scheme as well as the movement based scheme of [5] and a periodic checkpointing scheme such as in [9]. The interval between checkpoints in the scheme of [5] depends on the number of handoffs experienced by the MH. Each mobile host maintains a handoff counter and takes a checkpoint when the counter exceeds a threshold M. We use M as 3 which has been given as an optimal value of M in [5]. In any scheme using periodic checkpointing, such as in [9], the number of

Table 1. Log based recovery schemes for mobile computing systems.

Parameters		Proposed scheme	Movement based checkpointing & logging [5]	Optimistic logging based scheme [9]	Communication induced checkpointing and selective message logging [14]
Recovery related data Structure at the MH	Contents	<current BTS, previous BTS>	Complete set of BTSs with which the MH has registered since the last checkpoint	Complete set of BTSs with which the MH has registered since the last checkpoint	The BTS has a data structure, *hist*, which keeps track of the id of the BTS in which a given checkpoint of an MH is stored.
	Size	Less	More	More	Nil
MH can recover at a BTS different from where it crashed		Yes	Yes	Yes	Yes
Number of BTSs involved in recovery of an MH		Max 3 {IBTS, current BTS, previous BTS)	All BTSs in the log set , the number depends on mobility threshold	All BTSs in the log set	All BTSs in the array *hist* which may have some checkpoint related data of an MH
No of BTSs to which recovery related control messages sent		3	All BTSs in the log set	All BTSs in the log set	All BTSs in the system if the MH recovers in a different cell after failure

checkpoints depends on the frequency of checkpointing. We have used the time interval between checkpoints as 10 seconds for the simulation.

Since our scheme is a movement based scheme, our simulation experiments have compared the number of checkpoints taken by an MH with respect to the number of handoffs experienced by it. Till the time an MH is in a particular region, no checkpoints are taken. Only when there is a movement from one region to next, a checkpoint is taken. As a result of the experiments, we have observed that for MHs with less inter region movement, (*i.e.* those moving mostly between the BTSs of a region) there is a significant reduction in the number of checkpoints as compared to the other schemes. The scheme employed by [5] uses a threshold to initiate the checkpointing process. If an MH moves frequently, the checkpoints will be taken rapidly but if it does not move frequently, the size of the message log will increase at the BTS. Taking M as 3, we observe that the number of checkpoints increase linearly with the number of handoffs. Further, the number of checkpoints taken in the simulated periodic scheme is independent of the movement of the MH.

The simulation results are presented next. It can be seen in Fig. 3 that the number of checkpoints taken as per the proposed algorithm (with region size 4×4) is less than the number of checkpoints taken as per [5] with $M = 3$ or as compared to a periodic checkpointing scheme of [9]. Further, Fig. 4 demonstrates that as the region size is increased, the number of checkpoints taken by a mobile host decreases.

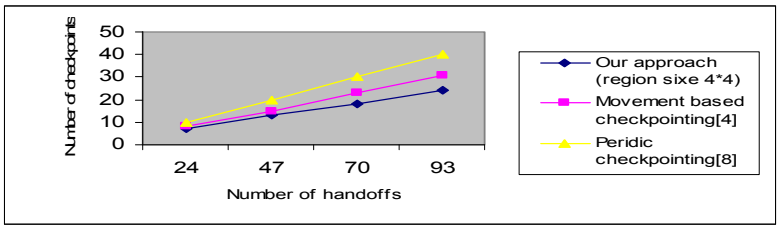

Fig. 3. Comparison of proposed approach with [5] and [9].

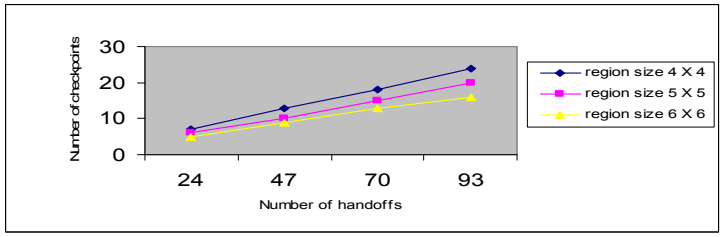

Fig. 4. Effect of increase in region size on the number of checkpoints.

7. CONCLUSION

In this paper, we have proposed a recovery protocol for the mobile computing environment which adds only a low overhead to the normal application execution. An MH can be located by passing messages only to the fixed hosts called the RCSs, which are fewer in number than the MHs, over the fixed media. A checkpoint can be taken only by initially passing message to the current BTS of the MH. A small number of messages are passed over the wireless media and the size of recovery related data stored at the MH is lower than in many other contemporary protocols. At any given time, it is required to keep only one checkpoint for an MH. The recovery algorithm does not lead to domino effect and a failed process needs to roll back to its latest checkpoint. In large networks with large region size, most of the movement of the mobile node will be within its region itself, thus reducing the number of checkpoints that need to be taken. This observation has been reinforced by the simulation performed.

REFERENCES

1. A. Acharya and B. R. Badrinath, "Checkpointing distributed applications on mobile computers," in *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems*, 1994, pp. 73-80.
2. B. R. Badrinath, A. Acharya, and T. Imielinski, "Designing distributed algorithms for mobile computing networks," *Computer Communications*, Vol. 19, 1996, pp. 309-320.
3. L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of ACM*, Vol. 21, 1978, pp. 558-565.
4. A. K. Singh and P. Kaur, "Log based recovery with low overhead for mobile computing systems," in *Proceedings of the 2nd International Conference on Advances in*

Communication, *Network and Computing*, 2011, CCIS 142, pp. 637-642.
5. S. E. George, I. Chen, and Y. Jin, "Movement based checkpointing and logging for recovery in mobile computing systems," in *Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, 2006, pp. 51-58.
6. N. Neves and W. K. Fuchs, "Adaptive recovery for mobile environments," *ACM Press*, Vol. 40, 1997, pp. 68-74.
7. H. Higaki and M. Takizawa, "Checkpoint recovery protocol for reliable mobile systems," in *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, 1998, pp. 93-99.
8. L. Alvisi and K. Marzullo, "Message logging: Pessimistic, optimistic, causal, and optimal," *IEEE Transactions on Software Engineering*, 1998, pp. 149-159.
9. T. Park, N. Woo, and H. Y. Yeom, "An efficient optimistic message logging scheme for the recoverable mobile computing systems," *IEEE Transactions on Mobile Computing*, 2002, pp. 265-277.
10. J. Ahn, S. Min, and C. Hwang, "A causal message logging protocol for mobile nodes in mobile computing systems," *Future Generation Computer Systems*, Vol. 20, 2004, pp. 663-686.
11. E. N. Elnozahy, L. Alvisi, Y. M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Computing Surveys*, Vol. 34, 2002, pp. 375-408.
12. D. B. Johnson and W. Zwaenepoel, "Sender-based message logging," in *Proceedings of the 17th International Symposium on Fault-Tolerant Computing*, 1987, pp. 14-19.
13. E. N. Elanozahy and W. N. Zwaenepol, "On the use and implementation of message logging," in *Proceedings of IEEE International Symposium on Fault Tolerance Computing Systems*, 1994, pp. 298-307.
14. T. Tantikul and D. Manivannan, "A communication-induced checkpointing and asynchronous recovery protocol for mobile computing systems," in *Proceedings of the 6th International Conference on Parallel and Distributed Computing*, *Applications and Technologies*, 2005, pp. 70-74.
15. R. Tuli and P. Kumar," Analysis of recent checkpointing techniques for mobile computing systems," *International Journal of Computer Science and Engineering Survey*, Vol. 2, 2011, pp. 133-141.
16. R. Garg and P. Kumar, "A review of fault tolerant checkpointing protocols for mobile computing systems," *International Journal of Computer Applications*, Vol. 3, 2010, pp. 8-19.

Parmeet Kaur Jaggi received B.E. in Computer Science and Engineering from P.E.C., Chandigarh, India in 1998 and M.Tech. in Computer Science from Kurukshetra University, India in 2008. She is currently working in Jaypee Institute of Information Technology, NOIDA, India. Her research interests include fault tolerance and checkpointing in distributed systems.

Awadhesh Kumar Singh received B.E. degree in Computer Science and Engineering from Gorakhpur University, Gorakhpur, India in 1988. He received M.E. and Ph.D. (Engineering) in the same area from Jadavpur University, Kolkata, India. He is an Associate Professor in Computer Engineering Department, National Institute of Technology, Kurukshetra, India. His current research interests include distributed algorithms, fault-tolerance, and mobile computing.