International Journal of Information Technology and Knowledge Management
July-December 2010, Volume 2, No. 2, pp. 657-664

Recent Advances in Checkpointing Techniques in Mobile Computing Systems

Prachi Maheshwari[1], Kamal Kant[2] & Lalit K. Awasthi[3]

This paper covers recent development that has taken place in fault tolerant techniques in mobile computing systems. A mobile computing system is a distributed system where some of the nodes are mobile computers. It consists of fixed node called Mobile Support station (MSS) and number of Mobile Hosts (MHs). A cell is a geographical area around a MSS in which it can support a MH. A Mobile Host can change its geographical position freely from one cell to another or even to an area covered by no cell. All the communication from one Mobile Host to another Mobile Host goes through MSS. MSS has both types of links- wired and wireless links. A MSS communicate with Mobile Hosts by wireless links, while with other MSSs by wired links. As the rapid advancement is taking place in mobile computing systems, so they are more vulnerable to faults. Fault tolerant techniques such as checkpointing have to be applied into these systems.
Keywords: Failure Rate, Checkpointing Interval, Global Checkpoint, Local Checkpoint, Active Interval.

1. Introduction

Checkpointing is a fault tolerant technique that allows system to roll back to a most recent failure free state when failure occurs in mobile computing system. Checkpoints are the snapshot of system state being taken during normal execution of a process. When system or process fails then it rolls back to last checkpoint state to achieve a failure free state. As mobile devices communicate with other mobile devices so at the time of recovery, rolling back of just one system which has failed may lead to inconsistency. So when a process rolls back to some previous failure free intermediate state, some other process on which failed process depends also roll back to achieve consistent global state. Checkpointing techniques are studied under uncoordinated, coordinated and communication induced checkpointing schemes. All these checkpointing scheme are further classified as shown in figure 1.

In this paper, a review of recent advancement in checkpointing scheme in mobile computing environment has been done.

The rest of the paper is organized as follows. In section 2, review of all the recent approaches in checkpointing in mobile computing system is done. Lastly we conclude this paper in section 3.

[1,2,3]Department of Computer Science & Engineering, NIT Hamirpur (H.P.), INDIA

Email: [1]prashi0110@gmail.com, [2]kamalkant25@gmail.com, [3]lalit@nitham.ac.in

2. Review Work

2.1. Uncoordinated or Independent Checkpointing

Each process can take checkpoint independently according to its convenience without coordinating with other processes in the system. A process can take checkpoint after a local timer expires or when the amount of information to be saved is small. It gives maximum autonomy to process to decide when to take checkpoint.

A scheme based on independent checkpointing and asynchronous message logging is proposed in [4]. All the messages are delivered to mobile host (MH) through MSS so message logs are saved by MSS for all MHs in its vicinity.

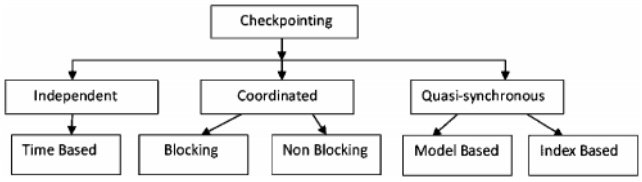
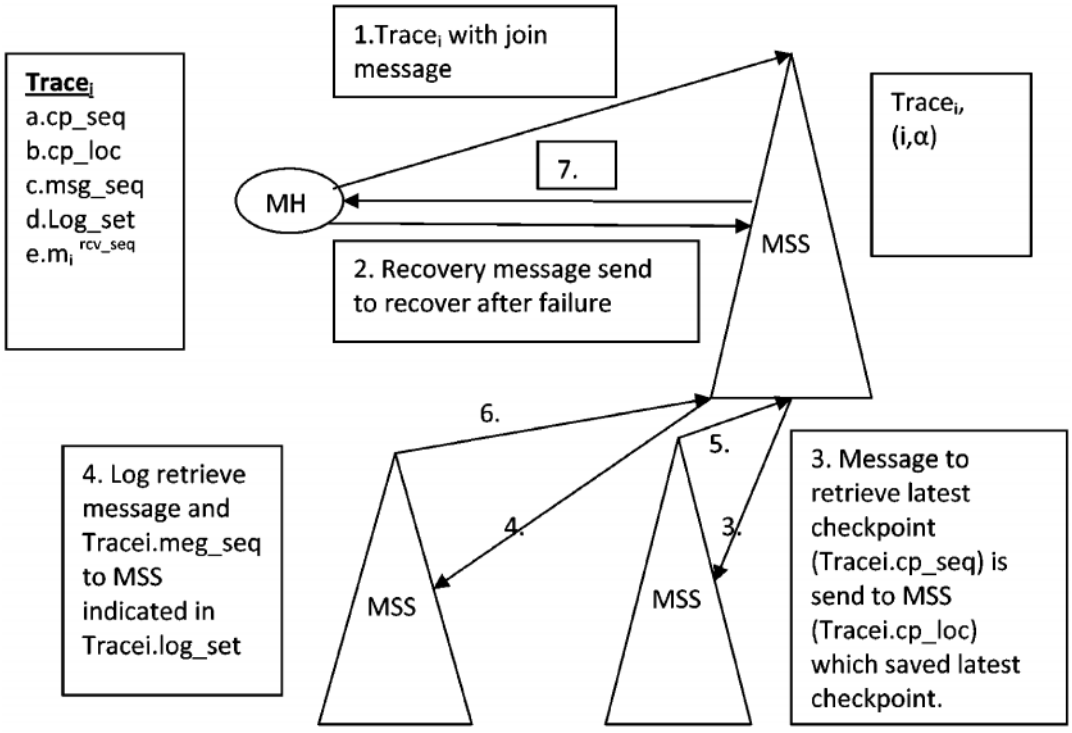


Fig.1: Classification of Checkpointing Schemes

These logs are used to recover state of process at MH after failure. The description of algorithm is shown in figure 2. When the mobility and failure rate of MHs is high then message logs of a MH may be distributed on number of MSSs which may be number of hops away from current MSS. Distributed storage at MSS need to be managed so as to reduce the cost of collection of message logs of MH after failure. Distance and frequency based scheme [5] allow movement of checkpoint and message logs to a nearby MSS when either distance between MH and MSS on which latest checkpoint is saved exceeds a threshold value, or when number of handoffs that is number of MSS carrying message

logs of a MH exceeds a threshold respectively. These schemes keep the recovery information of MH in certain range.

In these schemes, each MH stores Id of center MSS in its stable storage. Each MSS has a set of MSS which comes under its scope which is measured according to threshold distance between two MSS. When a MH moves between two cells, its recovery information is not required to be moved if a MH moves from MSS_p to MSSs belonging to scope of MSS_p. Disadvantage of distance and frequency based scheme proposed in [5] is that once a MH moves to another cell not in scope of center MSS of MH, all the recovery information of MH need to be migrated and if it returns back, partial recovery information may be unnecessary migrated. To overcome this disadvantage, another movement scheme is proposed in [6] in which when a MH moves out of a range, only partial recovery information of a MH needs to be migrated. It means recovery information of MH which is contained in some MSS due to mobility, is mapped to another set of MSSs. These MSSs are given by route function. It has advantage that one MSS is not overburdened by transferring all the information to it. Another movement based independent checkpointing scheme is proposed in [7] in which each MH maintain counter which is incremented by 1 when MH performs a handoff to other cell. Once this counter becomes greater than a predefined value, a checkpoint is taken. This counter depends on user's mobility rate, failure rate and log arrival rate. Each MH maintains set of MSS which stores MH's log after latest checkpoint. When a MH performs a handoff, a new MSS is added to this set if MH sends atleast one message in the cell of new MSS and if MSS has already not been added to the set. MSS logs messages before sending to destination. These messages are retrieved from MSS to recover a failure free state of MH after failure occurs. Once a new checkpoint is successfully taken by MH, set of MSS stored in MH is cleared and a message is sent to the MSS in the set to clear the log related to MH. Thus, storage management is done.



(i,α) is αth message delivered to MHi. log_set contains id of MSSs which have saved log for MH after latest checkpoint.

5. Reply with most recent checkpoint indicated by Tracei.cp_seq.

6. Messages logged with sequence number larger than or equal to received tracei.msg_seq (stores id of message received after latest checkpoint) are sent to current MSS for failed MH.

7. All the message log information received from all MSS and latest checkpoint are sent by current MSS to failed MH so that it can recover from failure.

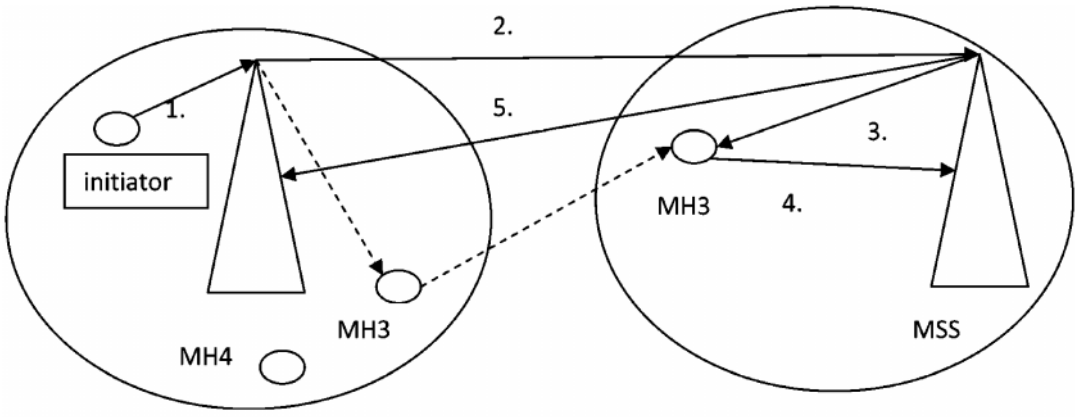
Fig. 2: Representation of Algorithm of Paper [4]

2.2. Blocking Checkpointing

In Blocking Coordinated Checkpointing, processes coordinate with each other in the sense that whenever a process called initiator takes checkpoint, it finds all the processes that have dependency relation with it. The initiator sends checkpoint request to all these processes when it takes checkpoint. All these processes are blocked from the time request for sending dependency vector comes to the time request for either discarding checkpoint or converting checkpoint to permanent checkpoint comes.

A two level blocking checkpointing algorithm is proposed in [8] in which local and global checkpoint are taken. Nodes take local checkpoint according to checkpoint interval calculated previously based on failure rate and save it in their local disk. These checkpoints when sent to stable storage become global checkpoint. Local checkpoints are used to recover from more probable failures where as global checkpoints are used to recover from less probable failures. After each checkpointing interval, system determines expected recovery time in case of permanent failure. System calculates amount of time taken (T1) to recover if system does not take global checkpoint and amount of time taken (T2) to recover if system takes global checkpoint. Then system compares these two times. If T2 < T1, system will take global checkpoint else system will only store checkpoint locally.

Another blocking coordinated scheme [9] is proposed in which each MSSp is required to maintain an array A[n] in which A[1] is 1 when MH1 is present in vicinity of cell of MSSp where number of MH (Mobile Host) are n starting from 0 to n-1. A MH initiates checkpointing procedure, calculates its dependency vector D and sends request to all the MH whose bit in dependency vector D is 1 via its MSS. If a MH is present in vicinity of current MSS, then checkpoint request is send directly to MH. Else current MSS will broadcast checkpoint request message to other MSS so that it can reach all those processes whose bit is 1 in dependency vector D calculated by checkpoint initiator. Thus all these processes take checkpoint and sends information to initiator via their local MSS. The description is shown in figure 3.



1. Initiator send coordination message M_{co} with dependency vector D to its MSSp
2. MSSp does not find MH3 in its cell so broadcast M_{co} with dependency vector to other MSSs
3. One MSS finds MH3 in its cell and thus sends checkpoint request message to MH3.
4. MH3 takes checkpoint and sends it to its current MSS
5. MSS thus forwards checkpoint with checkpoint sequence number to requesting MSS which saves this checkpoint and inform initiator.

Fig. 3: Pictorial Representation of Algorithm in Paper [9]

To reduce blocking time for checkpointing operation, Guohui Li and LihChyun Shu in [10] described an algorithm in which each process Pi maintains a set of processes Si. A process Pj is included in this set if Pj has sent at least one message to Pi in current checkpoint interval. Checkpointing dependency information is transferred from sending process to destination process during normal message transmission. So when a process starts a checkpointing procedure, it knows in advance the processes on which it depends both transitively and directly.

Another Scheme introduced in [11] uses the concept of Direct and Transitive Checkpoint Dependency. There are four data structures that are stored in stable storage of a MHi to maintain information about DCD and TCD. Tracei records identity numbers of MSSs that MHi has passed by. PROCESSEDi denotes whether MHi has received a

checkpointing request during global checkpointing procedure. RMi, SMi are received messages and send messages of MHi where current Mobile support station of MHi is MSSr. RMi and SMi are stored in a table at MHi. Figure 4 shows how the tables for RMi and SMi are maintained.

MHk sends message I' to MSSp and MSSp locates MHi's mobile support station MSSr. After inserting 2-tuple (i,p) into SMk, MSSp sends message to MSSr together with RMk and 2-tuple (k,p). When MSSr receives the message, it extracts the attached information, and then forwards it to MHi. In the proposed checkpointing scheme, when handoff of MHk occurs, handoff history is recorded in Tracek and no position changed messages are generated. When MHk leaves a cell, it sends leave message to old MSS say MSSp and then establishes a new connection by sending a connect message to new MSS say MSSr and it also sends old MSS id to new MSS. MSSr sends message to MSSp to fetch MHk's data structure. It inserts MSSp's identity number namely p into Tracek. If MSSr finds that number of items in Tracek has exceeds a predefined threshold, it will broadcast position changed messages to all MSSs in SMk. The threshold is used so that large number of messages are not sent if mobile host moves between the cell very frequently.

2.3. Non Blocking Checkpointing Algorithm

In Non Blocking Coordinated scheme, processes are not blocked during checkpointing and all processes coordinate among themselves to create consistent global state. There can be in-transit and orphan messages in the communication channel at the time checkpoint is taken. These messages are to be handled to avoid inconsistency in the system. Each message has a sending and receiving event. An orphan message is the message whose sending event is lost due to roll back but receiving event is saved at some process. A message is said to be in-transit if its sending event is saved at some process but its receiving event is lost.

Concept of active interval is used for handling orphan messages and lost messages in algorithm of [1]. Active interval is the time that elapses between two events of sending "prepare checkpoint" and "take checkpoint" message by initiator to all the processes. Messages are said to be lost if it is sent in active interval of a process P_i to P_j but received after active interval of P_j or not received at all. A message becomes an orphan message if it is sent by sender after its active interval and received by receiver before or in the active interval (AI). Three Counters are maintained – one for counting number of messages sent by a process i in its AI denoted by Sendi, two counters for receiving messages by process j before AI and in AI denoted by RMj_before and RMj_in respectively. By simple arithmetic (number of messages that are lost by process j = Sendi – (RMj_before + RMj_after))we can determine number of messages that are lost so that they can be replayed from sender side log. Orphan messages are handled by allowing receiver to maintain SSN for each message in its latest checkpoint which is count of number of messages received by the process uptil the last checkpoint.

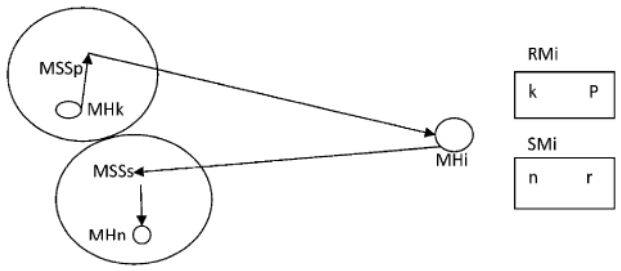


Fig. 4: Data Structures RMi and SMi for MHi (Mobile Host i)

If sender tries to replay any message whose SSN is less than or equal to SSN of receiver, receiver discard it as orphan message.

Awadesh kumar Singh in [2] proposed a non blocking algorithm in which a predefined checkpoint interval T is set on timers of all the MHs which is a deadline to take next checkpoint if process has sent any message in current checkpoint interval. Initially, every process sends a snapshot of its initial state (termed as 0th checkpoint) to its local MSS. A computation message is piggybacked with two values i.e. checkpoint sequence number Nj of sender Pj and local timer of MSS which is closest to receiver. Receiver sets its timer equal to this received time to achieve synchronization. On reception of computation message m by a process Pi, it checks its local timer has expired or not. If local timer has not expired, then it checks whether Nj of sender Pj is greater than Ni of receiver Pi. If it is true then Pi sets flag variable (If set to 0, process take checkpoint after expiry of its local timer otherwise checkpoint is not taken after expiry of local timer) to 1 and takes checkpoint if atleast one message is sent by process Pi in current checkpoint interval. Afterwards, Pi processes the received message m. As flag is set to 1 so Pi does not take checkpoint after expiry of its local timer. If Nj of sender Pj is not greater than Ni of receiver Pi, Pi processes the message m without taking checkpoint. Pi then takes checkpoint after expiry of local timer if it has sent atleast one message in current checkpoint interval. A global checkpoint consists of all the N^{th} checkpoints sent to stable storage by each process.

A non blocking algorithm that uses the concept of mobile agent to handle multiple initiations of checkpointing is proposed in [3]. Mobile Agent has id same as its initiator id and it migrates among processes, perform some work, take some actions and then moves to other node together with required information. Each process takes initial permanent checkpoint and sets version number of

checkpoint to 0. Process sends application message m by piggybacking it with version number of its latest checkpoint. Receiver compares application message's version number with its own current checkpoint version number to decide whether to take checkpoint first or simply to process message only. There is a DFS which is maintained by each process which contains id of neighbors on which the process depends. A number of initiator can initiate checkpointing at a time by creating a mobile agent for each initiation. Mobile agent follows DFS tree saved at initiator of all the neighboring nodes. Agent maintains 1) stack which has path to root, as required by DFS (depth first search tree) 2) visited list of processes which have been visited by this agent before any other agent in this checkpointing cycle. 3) partial list of concurrent initiators- mobile agent adds initiator of other agent into this list when it visit a process which has taken a checkpoint through another agent. It is list of processes who have initiated checkpointing. Agent before moving to neighbor, takes a temporary checkpoint at initiator and set checkpoint state to temporary, sets version number and adds initiator id into the its stack. It may visit three types of processes. It may visit a process whose checkpoint state is permanent that means this process has not taken a checkpoint in current checkpointing cycle; it takes a new checkpoint for this process and set checkpoint state to temporary, sets version number of checkpoint. It then pushes process into its stack, puts process in its visited list of processes. It may visit a process which has already taken a temporary checkpoint then agent checks whether checkpoint is induced by an application message. If it is true then agent sets it to false and adds process to its visited list of processes. If it visit a process where checkpoint is not induced by an application message and checkpoint state is temporary means current checkpoint is initiated by another agent from a different initiator. So process is added to its partial list of concurrent initiators. First phase of algorithm is completed when all the neighbors of initiator are visited. In the second phase, initiator with minimum id generates complete list of initiator. In the third phase, complete list of initiators is communicated to all other initiators by the initiator having minimum id. The initiator with minimum id will create two different agents, one for visiting the processes in visited list of processes and second agent for visiting processes in partial list of concurrent initiator. All the agents return to its creators and are destroyed at the end of third phase. Every process has one permanent checkpoint with same version number and this set establishes a consistent global state.

A non blocking scheme is described in [14] in which each MSS maintains vector Vi for process Pi in its vicinity. Vi tells about direct and indirect dependency. Vi = {V1i, V2i,…,Vni } where V1i is set to 1 if P1 has sent message to Pi since Pi's last checkpoint. When the normal computation message m is sent by process Pj to Pi, checkpoint sequence number (CSN) of sender is sent with message m, Pi compares its CSNi[j] with receivedCSN. If received CSN is greater than CSNi[j], it means Pj has already taken checkpoint beforesending m. So, Pi takes a forced checkpoint, increments CSNi[i], sets state of checkpoint as 1 which represents forced checkpoint, saves its local state to stable storage and then process m. If received CSN is less than or equal to CSNi[j], Pi processes m directly. Associated MSS of Pi sets Vji of Vi to 1. When a process Pi initiates a checkpointing procedure, it takes a tentative checkpoint, increments its CSNi[i]. Initiator sends a checkpoint request to each process Pj for which Vji is set in vector Vi and resumes its computation. When a process Pj receives a request from Pi, associated MSS of Pj checks vector Vj. If there is a process on which Pj depends but Pi does not depend, associated MSS of Pj sets Vi= Vi € Vj. When Pj receives checkpoint request message, it checks state of checkpoint. If it is 1, then Pj converts forced checkpoint to tentative. Else, Pj takes a tentative checkpoint, increments CSNj[j], sends copy to associated MSS.

The termination of algorithm is detected by initiator Pi when it gets a checkpoint completion message from all the processes for which Vi is 1.Then initiator broadcasts commit message to all the processes. All the processes will convert their tentative checkpoint to permanent checkpoint and old permanent checkpoint is deleted from memory.

In Mukesh Singhal and Guohong Cao's Algorithm [13], when a process takes mutable checkpoint, it does not ask other process to take checkpoint. Pi sends a message m to Pj by piggybacking CSNi[i] to m. Pj takes a mutable checkpoint if m CSN >= CSNj[i] and then process m. Otherwise, Pj process m before taking mutable checkpoint. These mutable checkpoints are converted to tentative checkpoint if process receives a checkpoint request. Mutable checkpoints aims at reducing overhead associated with coordinated checkpointing as it does not ask other processes to take checkpoint unless and until process itself gets checkpointing request from an initiator directly thus eliminating the "Avalanche effect". It is caused when a process recursively ask other processes to take checkpoints. These mutable checkpoints can be stored anywhere in main memory or disk. So, overhead associated with transferring large amount of data from MHs to MSSs over wireless links is reduced.

2.4. Communication Induced or Quasi Synchronous Checkpointing

It lies between synchronous and asynchronous (independent) checkpointing. Process takes communication induced checkpoints besides independent checkpoint to reduce number of useless checkpoints taken in independent checkpointing approach. Processes takes two kinds of checkpoints called local checkpoints and forced checkpoints. Local checkpoints are just like independent

checkpoints taken in independent checkpointing approach. Forced checkpoints are taken to guarantee eventual progress of recovery line. During normal operation, checkpoints are taken normally but when failure happens then a recovery line is found to determine consistent global checkpoint among multiple checkpoints taken by each process.

Jin Yang, Jiannong Cao, Weigang Wu in [12] proposed a communication induced checkpointing scheme in which communication induced or forced checkpoints are taken by a process by analyzing piggybacked information that comes with received message. Each process has a logical clock or counter which is increased with every new checkpoint taken. When a process sends an application message, it piggybacks recent value of logical clock on message. Receiver compares its LC (logical clock) with received LC to decide whether to take a forced checkpoint before processing message or simply process the message.

Algorithm uses a Mobile Agent (MA) system which has a globally unique id. Each MA executes on a node and takes an independent checkpoint before migration. It then determines next host to which it has to migrate, it reaches on that host and takes a checkpoint on it. This process will continue until all hosts have been visited. These checkpoints are called local checkpoints.

The basic checkpoints that are taken can act as independent checkpoint if Basic CIC algorithm described before is integrated with reliable migration algorithm. It can be done by letting each MA take basic checkpoints according to frequency defined by algorithm of reliable migration. In Basic CIC, when a process receives a message m from another process then if condition described before is true, then a forced checkpoint is taken. Deferred message processing scheme is used to prevent a mobile agent of a process from taking forced checkpoint on receiving of a message in Basic CIC. MA only accepts message and store it in MA's associated mailbox (buffer). The processing of received message is delayed until MA takes a basic checkpoint and lands on new host. Thus forced checkpoints are avoided but the price is that messages which can result in forced checkpoint are not processed immediately.

3. CONCLUSION

We have reviewed and compared different approaches for failure free execution of a mobile host and to a greater extent failure free execution of mobile environment. We studied three checkpointing scheme- independent, coordinated and communication induced checkpointing. We have also compared different approaches of checkpointing and determined their advantages and disadvantages in table 1.

Table 1
Comparison of Different Approaches of Checkpointing

Ref. no., publication year	Techniques used	Advantage and disadvantages
[1], 2008	Concept of Active interval used to handle orphan and lost message.	1. Easier scheme to determine lost and orphan messages.
[4], 2002	A scheme based on independent checkpointing, optimistic and asynchronous message logging is proposed	1. Recovery of an MH is performed independently. 2. Message logging & dependency tracking is performed by MSS to utilize volatile space at MSS.
[2], 2007	Non blocking algorithm in which a predefined checkpoint interval T is set on timers of all the MHs initially. A MH takes checkpoint only when its local timer expires and if it has sent any message in current checkpoint interval.	1. Tracking of dependency information is not required. 2. Checkpoints taken only when process has sent at least one message in the current checkpointing interval, no temporary checkpoint taken. 3. Timer used to indirectly coordinate creation of consistent global State. 4. Techniques for synchronization of timers are to be used.
[3], 2007	A scheme to handle multiple initiations of checkpointing is proposed. DFS tree saved at initiator is used to find neighbor node that has to be visited next by mobile agent. It uses the concept of mobile agent.	1. Total number of moves required by all the agents in complete checkpointing process may be very large. 2. Multiple concurrent initiation of checkpointing process is possible.
[5],2003	Algorithm for managing distributed storage of message logs of MH when a mobile host is saving message logs at these MSSs is described. Distance and frequency based movement scheme are defined.	Movement based schemes used to handle distributed storage, controls transfer cost as well as recovery cost.

Contd...

Contd...

Ref. no., publication year	Techniques used	Advantage and disadvantages
[11], 2005	They introduced the concept of Direct Checkpoint Dependency (DCD) and Transitive Checkpoint Dependency (TCD).	1. Dependency information, MH's location information are maintained in corresponding MSS instead of MH, which save limited power and storage of MH. 2. MSS will broadcast position change message for MH if number of MSS through which MH has passed exceeds a threshold.
[14], 2007	Non blocking algorithm uses communication vector to identify direct and transitive dependency.	Only few processes take checkpoint when a initiator initiates checkpointing procedure.
[8],2009	A two level blocking checkpointing algorithm in which local and global checkpoint are taken, is proposed.	Algorithm has lower overhead than one level scheme as local checkpoints are used for more probable failures and global checkpoints are used for less probable failures.
[6], 2008	Recovery information of MH which is contained in some MSS due to mobility, is mapped to another set of MSSs when distance between MH's current MSS and its old MSS exceeds a threshold. These MSSs are given by route function.	1. Independent checkpointing algorithm. 2. Failed nodes can recover independently. 3. All the recovery information of a MH is not transferred to single MSS, so one MSS will not be bottleneck of failure and access.
[12], 2006	1. Communication induced checkpointing scheme. 2. Processes take two kinds of checkpoint: a. Basic (independent) checkpoints b. Forced checkpoints taken to maintain consistent recovery line.	Deferred message processing scheme allow delaying the processing of received message (that can lead to forced checkpoint) until mobile agent takes a basic checkpoint. Thus forced checkpoints are avoided but some messages cannot be processed immediately.
[7], 2006	Independent checkpointing and optimistic message logging used. A MH takes checkpoint when its handoff_counter becomes greater than a predefined threshold.	1. When a MH crosses a distance threshold from the location of latest checkpoint, recovery information is collected and transferred to MH's local MSS.2. Storage management is done by removing the log entry from MSS when a checkpoint is successfully taken.
[16], 2006	Application programmer places checkpoint request in the program at the point when it is most efficient and system may grant or deny the request based on various heuristics.	1. Both application programmer and runtime system takes decision when to take checkpoint. 2. It is used in many software developing companies. 3.Cooperative Checkpointing scheme.
[9], 2007	A Blocking coordinated Checkpointing scheme is described.	Issue related to Mobility management is addressed. A solution to how mobility can be handled in blocking coordinated scheme is given.
[10], 2005	A blocking coordinated checkpointing scheme is defined which forces only minimum number of processes to take checkpoint when a process initiate checkpointing.	Process knows in advance about all the processes on which it depends both transitively and directly, thus blocking time is reduced.
[13], 2001	Concept of mutable checkpoint is used. These are neither tentative nor permanent checkpoint but can be turned into tentative when request for taking checkpoint comes.	Process taking mutable checkpoint does not ask other processes to take checkpoint unless and until other processes itself get checkpointing request from an initiator directly thus eliminating the "Avalanche effect".
[15], 2007	When a process has good probability of receiving checkpoint request, then induced checkpoint is taken before processing message. If probability is not good, then process buffers message till it takes checkpoint or receives commit message.	1. Tentative minimum set of processes calculated and made available in beginning to all the MSSs so as to reduce blocking time. 2. Useless checkpoint is reduced.3. Selective message are delayed at receiver end so to allow process to send message during its blocking period.

REFERENCES

[1] Ch. D. V Subha Rao and M.M Naidu "A New Efficient Coordinated Checkpointing Protocol Combined with Selective Sender based Message Logging", 2008.

[2] Awadhesh Kumar Singh, "On Mobile Checkpointing using Index and Time Together" World Academy of Science, Engineering and Technology, 2007.

[3] Partha Sarathi Mandal and Krishnendu Mukhopadhyaya, "Mobile Agent based Checkpointing with Concurrent Initiations", International J. of Foundation of Computer Science, 2007.

[4] Taesoon Park, Namyoon Woo and Heon Y. Ycom, "An Efficient Optimistic Message Logging Scheme for Recoverable Mobile Computing Systems", IEEE Tran. on Mobile Computing, 2002.

[5] Taesoon Park, Namyoon Woo and Heon Y. Yeom, "An Efficient Recovery Scheme for Fault Tolerant Mobile Computing Systems", FGCS- 19, 2003.

[6] Yi-Wei ci, Zhan Zhang, De- Ching Zuo, Zhi- Bowu and Xiaa-Zong Yang, "Area Difference Based Recovery Information Placement for Mobile Computing System", 14 th IEEE international Conference on Parallel and Distributed Systems, 2008.

[7] Sapna E. George, Ing-Ray Chen and Ying Jin "Movement Based Checkpointing and Logging for Recovery in Mobile Computing Systems", ACM, June 2006.

[8] Mehdi Lotfi, Seyed Ahmad Motamedi and Mojtaba Bandarabadi, "Lightweight Blocking Coordinated Checkpointing for Cluster Computer Systems", Sym. on System Theory, 2009.

[9] Suparna Biswas and Sarmistha Neogy "A Low Overhead Checkpointing Scheme for Mobile Computing Systems", Int. Conf. Advances Computing and Communications, IEEE 2007.

[10] Guohui Li and LihChyun Shu "A Low-Latency Checkpointing Scheme for Mobile Computing Systems" Int. Conf. Computer Software and Applications, IEEE, 2005.

[11] Guo-Hui Li and Hong-ya Wang, "A Novel Min Process Checkpointing Scheme for Mobile Computing Systems", Journals of System Architecture, 2005.

[12] Jin Yang, Jiannong Cao and Weigang Wu, "CIC: An Integrated Approach to Checkpointing in Mobile Agent System", Proceedings of Second International Conference on Semantics, Knowledge and Grid, 2006.

[13] G.Cao and M.Singhal "Mutable Checkpointing: a New Checkpointing Approach for Mobile Computing System", IEEE Trans. Parallel Distributed System, 2001.

[14] Men Chaoguang, Cao Liujuan, Wang Liwen and Xu Zhenpeng , " Low Overhead Non Blocking Checkpointing Scheme for MCS", Tsinghua Science and Technology, July 2007.

[15] Lalit Kumar Awasthi and P.kumar "A Synchronous Checkpointing Protocol for Mobile Distributed System: Probabilistic Approach", Int. Journal of Information and Computing Security, 2007.

[16] Adam Oliner, Larry Rudolph and Ramendra Sahoo "Cooperative Checkpointing Theory", IEEE, 2006.