Analysis of Checkpointing and Rollback Recovery Algorithms in MANET

Ms. Remya K
M. Tech Student
Department of Computer Science & Engineering
Thejus Engineering College, Thrissur, India

Abstract— Trends in high-performance computing are making it necessary for long-running applications to tolerate hardware faults. Checkpoints are used to store the specific states of processes in stable storage. When systems fail, they can recover to a consistency state from checkpoints rather than the beginning state. There are many checkpointing algorithms used in distributed systems. However, the checkpointing algorithms proposed for traditional distributed systems cannot be used in MANETs due to its unique characteristics such as low storage capacity, limited bandwidth, frequent disconnection, etc. Checkpointing technique can be basically classified into three categories that are: uncoordinated checkpointing, coordinated checkpointing and hybrid checkpointing. Uncoordinated protocols allow different nodes to save their local states independently of each other. In the coordinated protocols, a node coordinates the checkpointing. Hybrid checkpointing is the combination of two or more checkpointing algorithms. The aim of this paper is to analyze various checkpointing algorithms and its applications in MANETs.

Key words: Checkpointing, Stable storage, Coordinated checkpointing, Uncoordinated checkpointing

I. INTRODUCTION

A mobile ad hoc network (MANET) is an autonomous collection of mobile nodes that communicate through wireless links. The network topology is dynamic and decentralized. Nodes can move freely in the network. All the network activities including discovery of topology and delivery of messages must be executed by the nodes themselves. They develop through self-organization. Due to the communication of nodes over wireless links, they have to contend with the effects of radio communication such as noise, fading and interference. In addition, the links typically have less bandwidth than in a wired network. The nodes have limited storage capabilities and typically no stable storage. The battery life determines the lifetime of a node.

In such a scenario, the failure probability of the computing process increases greatly along with enlarging scale of the system. If a failure occurs in a computing process and there is not an appropriate method to protect it, more cost will be wasted for restarting the program. This need for reliability leads to the requirement of some fault tolerance method specifically designed for such networks. A major class of distributed systems uses checkpointing along with rollback recovery for providing fault tolerance. Clustered MANETs in which nodes are partitioned into a number of disjoint groups called clusters are considered here. Under the cluster structure, mobile nodes are assigned a different function, such as cluster head (CH) or cluster member. One node in each cluster is chosen as the cluster head based on some criteria and the other members of the cluster use the stable storage at the cluster head for saving their checkpoints. A cluster head is a local manager of all mobile hosts within a cluster. One of the basic functions for a cluster head is broadcasting beacon packets to all mobile hosts in the cluster. The clustering structure reduces routing control overhead and improves the network scalability.

As fault-tolerance is an important design issue in building a reliable ad hoc network, MANETs must be fault tolerant that is they must be able to recover even after a failure occurs. Transient failures in system are the one which stays for short duration time during operation only. The fault tolerance technique helps the system to resume the computation from the last consistent state and thus reducing the recovery time. There are various recovery schemes that have been proposed to make the system fault tolerant such as log based recovery, rollback recovery and checkpointing.

Checkpointing is a technique for fault tolerance in computing systems. It basically consists of taking a snapshot of the current application state, storing it on some memory area and later on using it for restarting the execution from that particular point in case of failure. The normal processing of a process is interrupted specifically to preserve the status information necessary and then to allow resumption of processing at a later time. Computation may be restarted from the current checkpoint instead of repeating it from the beginning if a failure occurs.

Various types of checkpointing protocols have been proposed in the literature: synchronous checkpointing, where each process checkpoints simultaneously with every other process [1], quasi-synchronous checkpointing, where the communication history is piggybacked on each message, and each process checkpoints independently based on that information, and asynchronous checkpointing, where each process checkpoints independently, but the end result may be an inconsistent global state. However, none considers contention.

Local checkpoint is the saved state of a process at a processor at a given instance. Global checkpoint is a collection of local checkpoints, one from each process. A global state is said to be “consistent” if it contains no orphan message; i.e., a message whose receive event is recorded, but its send event is lost. Initial global state of a process is always consistent, because it cannot contain any orphan message. A transit message is a message whose send event has been recorded by the sending process but whose receive event has not been recorded by the receiving process.

II. RELATED WORK

The fault tolerance schemes for the distributed systems cannot be applied directly in ad hoc wireless networks due to the reason that there is no support of any static centralized

administration and there are no fixed stable hosts or mobile support stations.

Biswas and Neogy[2] proposed mobility aware checkpointing and failure recovery algorithm for cluster based MANETs where the mobile node saves checkpoints in neighboring nodes if the mobility of a node among the clusters crosses the threshold value and in case of failure recovery of node done through the mobile cluster head. This algorithm shows the minimum checkpoint and log overhead per mobile host per checkpoint interval and no orphan/lost messages.

Jaggi and Singh [3] introduced a concurrent checkpointing and recovery scheme. They presented a staggered approach in their work to avoid simultaneous contention for resources. The events which would normally happen at the same time are forced to start or happen at different times by staggering. This protocol logs minimum number of messages and does not need any FIFO channels. It successfully handles the overlapping failures in ad hoc networks and supports concurrent initiation of checkpoints. This approach is specially required for synchronous checkpoint where the number of processes taking their checkpoints simultaneously is more hence the checkpoint size grows, brings huge benefit to system performance.

Men et al. [4] presented checkpointing and rollback recovery scheme for the cluster-based multi-channel ad hoc wireless network management where the cluster head controls the MHs to take checkpoints in checkpoint beacon intervals and to rollback to consistent state in case of failure. Every beacon interval consisting of different phases depicts for checkpointing and recovery scheme capable of handling ordinary host transient failures and crash of gateway between two neighbor clusters. CH uses beacon packet which contains clock data, traffic indication messages and data window and also holding variables such as checkpoint index, ordinary node queue and reply messages. The recovery scheme has no domino effect and the failure process can start from its latest local consistent checkpoint, then replay the messages and to make gateway consistent state, repeated message for rollback will be discarded. The simulation results show this scheme keeps fast recovery upon transient failures and only a low additional overhead is incurred.

Tuli and Kumar[5] presented asynchronous checkpointing and optimistic message logging for mobile ad hoc networks as in the unreliable mobile host and fragile network connections, only checkpointing scheme will not suffice to address the problem correctly, hence message logging is also included which carried out by CHs. In this scheme each MH takes checkpoint independently and messages delivered to MH routed through CH which logs the message on its own stable storage and avoids the overhead message logging to MHs. This algorithm copes with mobile host failures and disconnections from cluster, especially the hand off procedure with sequence of events helps in recovery of information transfer.

Saluja and Kumar[6] in their work discussed a new minimum process checkpointing procedure for mobile ad hoc networks which is based on the cluster based routing protocol that reduces routing traffic and prohibitive of flooding traffic in discovery of routes. Any process (MH) can initiate checkpoint in this algorithm, it takes tentative checkpoint before sending message and then send request to CH, and now CH on behalf of MH coordinates checkpointing operation with other processes. Only those process participate in checkpointing operation with the initiator which are present in the minimum processes set created with Z-dependencies notion. This algorithm ensures that blocking of processes does not take place and takes no useless checkpoints as it maintains exact dependencies and piggybacks checkpoint sequence number, dependency vector onto the normal message communication.

Juang and Liu [7] approach facilitates independent checkpointing and rollback recovery technique in multihop communication MANET. In the state transition interval called interval index depends message received by the process and state of process, finally spurs up the dependency matrix considering both transitive and direct dependencies. All the communication is transmitted from cluster to cluster goes through the cluster head node CH. Dependency matrix and message logs maintained by CH hence no additional overhead to MH and also this scheme covers resending of lost messages when process fails.

Jaggi and Singh [8] presented self-stabilizing spanning tree applied on dynamic network topology i.e. clustered and multi-hop mobile ad-hoc network reduces the message overhead and controls dynamic nature of MANETs. Snapshot initiation can be done with multiple initiators but only Leader CH initiates, sends message to the other MHs, eventually receives snapshot from all its child nodes and computes global states as set of local states in non-FIFO channels. The CHs are organized into a self-stabilizing spanning tree to reduce the number of snapshot related messages as compared to the approach of broadcasting the messages along each outgoing channel in traditionally way. This algorithm shows better performance large scaled cluster system as it needs less number of control messages to hold self-stabilizing spanning tree.

III. METHODOLOGY

A. System Model

In large dynamic MANET, a link based proactive routing scheme cannot perform well. One simple approach for communication is called packet flooding. When both the communicating nodes are in host's radio coverage, they can communicate directly. Otherwise a route is required to forward messages to the destination in a multi hop manner that cause the packet flooding.

A Cluster based or hierarchical architecture is proposed to reduce the flooding packets and to minimize the routing table. Clustering approach in an ad hoc network group its nodes into many clusters. In each cluster one node act as a head node (CH) and rest of nodes are divided into ordinary and gateway nodes. The cluster head is responsible for communication within the ordinary nodes in own cluster and gateway node is responsible for in between two adjacent clusters. A cluster is characterized by two types of messages – inter and intra-cluster message. Since Normal nodes only communicate with their cluster head, which in turn, aggregates the collected information and sends it to the MSS. In this scheme, cluster head failures are critical. When it fails, re-election process is invoked within the cluster.

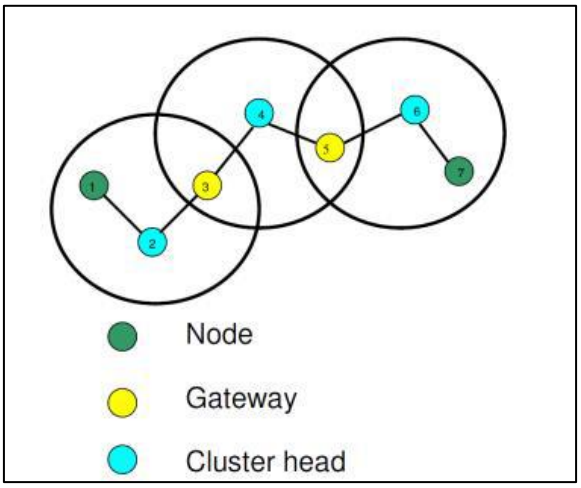

Fig. 1: Clustering Architecture in MANET

B. Uncoordinated Checkpointing

Uncoordinated protocols allow different nodes to save their local states independently of each other. Although, no additional messages due to checkpointing are generated, multiple local checkpoints should be performed. During the rollback, the recovery line is computed using timestamps and based on causal dependencies. The uncoordinated checkpointing aims to maximize the performance for normal operation by minimizing the overflow of control messages. The drawbacks of this technique are: the risk of the domino-effect (Cascading rollback to initial state); some checkpoints are not useful (not belong to any consistent global checkpoint) and SS capacity must be sufficient. Uncoordinated checkpointing allows any process can initiate checkpointing. The advantage of this scheme is that each process may take a checkpoint in any critical state.

1) The Algorithm Concept

A staggering based synchronous checkpointing scheme is an uncoordinated checkpointing scheme adapted for handling the limited storage and bandwidth problems of MANETs. Controlled sender based message logging is used, where only those messages are recorded in the message log that have been sent and yet not received at their respective destinations. They are in-channel messages because these are the messages presently in the channel. The messages are given sequence numbers to maintain consistency at the time of recovery. The staggering causes events, which would normally happen at the same time, to start or happen at different times. Checkpoints are initiated concurrently and such initiation handles the overlapping failures in ad hoc networks.

Each CH has a distinct identifier (or id) and the hosts on joining a cluster also get an id. Each CH keeps a list, LOCAL of the active nodes in its cluster. The nodes in a cluster use the stable storage at the CH to save their most recent checkpoints. Any CH may initiate the checkpointing and there can be multiple initiators of the checkpoint process. The initiator CHs are called leaders. The checkpoints are assumed to be sequenced so that all checkpoints with the same sequence number form a consistent state of the system.

A leader CH after taking its own checkpoint sends a take_chkpt message, containing its own id, to other CHs in its transmission range and then initiates the checkpoint in its cluster. All the clusters which take a checkpoint in response to the message from the same leader form a group. Thus there will be as many groups formed in the system as there are concurrent leaders. Any CH on receiving the take_chkpt message for the first time saves the sender's id as its PARENT and the initiator's id as its LEADER. Every CH keeps a record of the recipients of its take_chkpt message by a 2D array, GLOBAL. A CH on receiving a take_chkpt message another time, sends a DENY message to the sender. It however keeps track of any concurrent leaders using a boundary-set data structure. If the initiator's id in any subsequent take_chkpt message is different from that saved in the LEADER variable at the CH, then the receiver CH saves this initiator's id in its boundary-set after sending a DENY message to the sender CH.

Thus the CHs will form a forest of spanning trees in the system. Each leader is the root of a spanning tree and all CHs which take a checkpoint due to it belong to its spanning tree. A CH which sends a take_chkpt message to another for the first time is its parent in the spanning tree. The receiver of this take_chkpt message is its child node. Within a cluster, a CH, after taking its own checkpoint, initiates the checkpoint by sending the take_chkpt message to its cluster nodes one by one in increasing order of their ids. This procedure continues till the last member of the cluster. Hence the nodes of the cluster take checkpoints at the CH in a staggered fashion. When a leaf CH in the spanning tree has completed a checkpoint in its cluster, it sends an ACK message along with the boundary set to its parent in the spanning tree. This boundary-set is merged with the parent's boundary-set. After an intermediate CH in a spanning tree has received such ACK messages from its entire set of child CHs and has completed the checkpoint in its cluster, it sends an ACK message along with the boundary-set to its parent in the spanning tree. When the leader receives the acknowledgement of all its children CHs, it also knows the identifiers of other initiators in the system using boundary-set information it receives from the child CHs. The initiator then sends the chkpt_taken message to other initiators. When it has received similar messages from all concurrent initiators, it propagates a chkpt_taken message in the group formed by its child nodes to complete the checkpointing process for a given sequence number.

2) The Recovery Procedure

The messages in the channel are logged at the sender. Fig. 3 shows a failure at process Pi after it has sent messages m & m' to process Pk and received the message m" from Pj since its last checkpoint i2. If the process Pi needs to recover after the failure, it rollbacks to its latest checkpoint, here i2, and replays the logged messages.

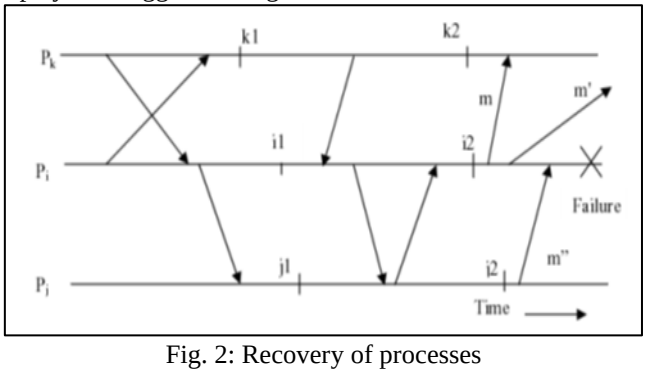

Fig. 2: Recovery of processes

Hence any process Pk that has received messages from Pi before Pi's failure need not rollback as it has either already received the message m from Pi before Pi's failure or will receive the in-channel message m' at the time of Pi's recovery. However, if a process Pj has also sent messages to Pi then Pj needs to rollback to the last checkpoint , here j2, so that it can re-send any message m" earlier sent by Pj but un-received due to the rollback by Pi. To avoid inconsistency in the system due to the recovery process, two approaches are possible. A non-blocking approach would allow processes like Pk to continue normal operation during recovery of any other process Pi to which no messages were sent during the last checkpoint interval. In such a case, all checkpoints of the same sequence number would form a consistent state of the system. A second blocking approach for recovery can require a recovering process Pi to send a RECOVERING message to all processes in SentSeti so that they do not advance their checkpoint till Pi has recovered. This will prevent orphan messages in the system. Upon the completion of recovery process, Pi can send a RECOVERED message to SentSeti. In the non-blocking approach, no process is restarted from a state that has recorded the receipt of a message that no other process has recorded as received. If a process Pk has recorded the receipt of a message m between its checkpoint k2 and a later checkpoint k3, the sending of m shall be recorded by Pi between its checkpoint i2 and a later checkpoint i3. The recovery procedure leads to a consistent state in the system.

C. Coordinated Checkpointing

In the coordinated protocols, a node (initiator) coordinates the checkpointing. This technique simplifies the recovery, does not lead to a domino-effect and reduces memory overflows on SS but however induces a higher latency. Some coordinated protocols need to block the communication during the coordination. The MANET structure used in this algorithm is hierarchical based. The scheme is based for Cluster Based Routing Protocol (CBRP) which belongs to a class of Hierarchical Reactive routing protocols. The protocol is nonblocking coordinated checkpointing algorithm suitable for ad hoc environments. It produces a consistent set of checkpoints; the algorithm makes sure that only minimum number of nodes in the cluster is required to take checkpoints; it uses very few control messages.

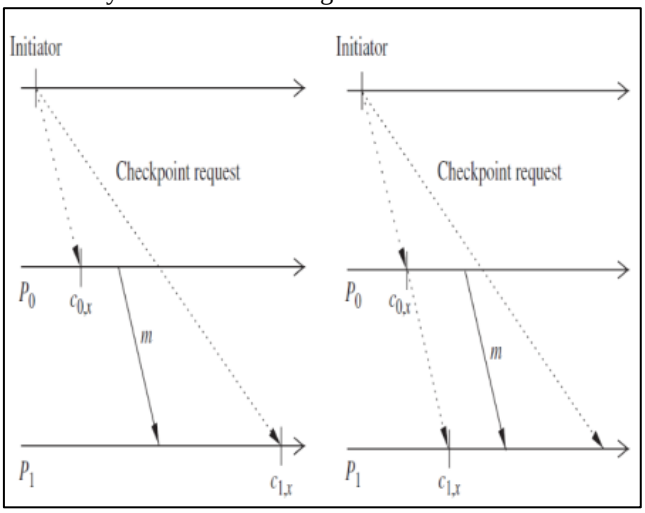

Fig 3: Coordinated checkpointing

1) Blocking Coordinated Checkpointing

- Phase 1: A coordinator takes a checkpoint and broadcasts a request message to all processes, asking them to take a checkpoint.
- When a process receives this message, it stops its execution and flushes all the communication channels, takes a tentative checkpoint, and sends an acknowledgement back to the coordinator.
- Phase 2: After the coordinator receives all the acknowledgements from all processes, it broadcasts a commit message that completes the two-phase checkpointing protocol.
- After receiving the commit message, all the processes remove their old permanent checkpoint and make the tentative checkpoint permanent.

2) Non-blocking Checkpoint Coordination

The objective of the coordinated checkpointing is to prevent a process from receiving application messages that could make the checkpoint inconsistent. Checkpoint coordinator / initiator broadcast the checkpointing message to every other node. Each node upon receiving this message should take a checkpoint. However, this approach could lead to checkpoint inconsistency.

D. Hybrid Checkpointing

The context deals with a distributed application that is assumed to run on N mobile nodes (or MH: Mobile Host). Moreover, the global system is assumed to be asynchronous: it has neither a global clock nor a shared common physical memory. Nodes communicate by exchanging messages; this is the only way to synchronize between MHs. Each process (hosted on an MH), runs at its own speed. Messages are assumed to be exchanged through bidirectional and reliable channels. In addition, messages are not lost or corrupted during transmission. In the Class of Quasi-synchronous protocols, autonomy of nodes is preserved (as it is in the case for the class of independent checkpointing), making it possible to engage arbitrarily a spontaneous (voluntary) checkpointing process. Such a technique exploits the concept, called communication induced (type of synchronization) in order to make progressing the consistent recovery line and preserving the autonomy of MHs by making them taking checkpoints asynchronously. This kind of algorithms enjoys no overflow of control messages and guarantees the existence of a consistent recovery line. However, this approach suffers from a large number of checkpoints, thus yielding a high computation complexity when processing the rollback-recovery operation. Consequently, such a solution, as it was defined, seems to be very greedy and inappropriate for MANETs.

In order to adapt the previous solution so that to be applicable in MANETs, we should reduce the number of messages in the system, and avoid that MHs record "all" their checkpoints on the stable storage (SS). Hence, we propose to save checkpoints locally and engage another process that requires a general synchronization to save the permanent checkpoint on the SS. Assuming that, we propose in the sequel a hybrid checkpointing algorithm that performs in two phases: the quasi-synchronous phase and the coordination phase. The first phase of the algorithm allows to each MH to progress independently and take a spontaneous checkpoint

and save it locally. This is more appropriate for MANET, because it limits the data transfer into SS, thereby reducing the energy and the bandwidth consumption. The second phase is dedicated to coordination (synchronization). An MH coordinator initiates the recording of the global checkpoint. It broadcasts the checkpointing request to its direct neighbors, which propagate it to their own neighbors, and so on until covering all the nodes of the network. At the end of this phase, the different checkpoints, already saved locally, are transferred to SS to determine one consistent global permanent checkpoint.

In other respects, the protocol should adjust its functioning according to the environment and the nature of the underlying distributed computation. Within this intention, two parameters are initialized at the beginning of the computation:

- α: Denotes the maximum number of checkpoints allowed for an MH to take locally before any coordination.
- T: corresponds to the maximum time period observable between two spontaneous checkpoints.

1) Description of the checkpointing protocol

a) Phase 1 (Quasi-Synchronous):

Each node i is associated with two integer variables (Sni and Nexti initialized to 0): Sni is the number in the sequence of the last checkpoint taken locally; Nexti is a counter incremented every T time units making it possible to know whether a spontaneous checkpoint can be taken. Person re-identification means it's a process of identifying a person which appears once in a camera of any public places. The person re-identification methods can be done in various ways in different methods like image based and video based person re-identification methods. One of the major challenges in person re-identification method is the lack of spatial and temporal changes. The Comparative Attention Network (CAN), Accumulative Motion Context (AMOC), Person re-id with Reference Descriptor, PersonNet and Temporally Memorized Similarity Learning based person re-identification methods are discussed here.

During this phase, each node i take a checkpoint and save it locally. The node i maintain a stopwatch which is set initially to T. When the timeout expires, it takes a spontaneous checkpoint if it was not compelled during the time interval to take a forced checkpoint. Hence, in this phase, two kinds of checkpoints can be saved locally:

- Spontaneous checkpoint: The node i takes it every T time units (after incrementation of Nexti) if the value of Nexti > Sni. In other terms, if the node i did not take a forced checkpoint.
- Forced checkpoint: The node i must take a forced checkpoint in two cases: (i) The value of Sn piggybacked in a computation message M, denoted by M.Sn, is greater than the local value Sni: (M.Sn > Sni). (ii) At the reception of a checkpointing request CP Request with a value of Sn greater than the local value Sni: (CP Request.Sn > Sni).

The example below depicts the two types of checkpoints.

Example 1: We have three nodes (as shown in Fig. 4): MH1, MH2, and MH3 that exchange messages. Spontaneous checkpoints are taken periodically and represented by' ['. Forced checkpoints are represented by' [*'.The sequence numbers corresponding to the checkpoints are indicated by Sn. The value of Next is incremented every T time units. The message M sent by MH1, within which is piggybacked the sequence number (Sn = 5) > (Sn3 = 4), forces MH3 to take a forced checkpoint before processing the message. During the next period, MH3 is discharged of taking a spontaneous checkpoint.

b) Phase 2 (Coordination):

While checkpoints are taken, each node i checks whether its sequence number (since the last recorded permanent checkpoint), has not reached the predefined threshold α (by comparing values of Sni − Cp Request.Sn and α). When the threshold is reached for a node i, it initiates the second phase of coordination. This phase consists, first, in collecting all the checkpoints having the same sequence number (Sni) to determine a consistent global permanent checkpoint, and then storing it into the SS.

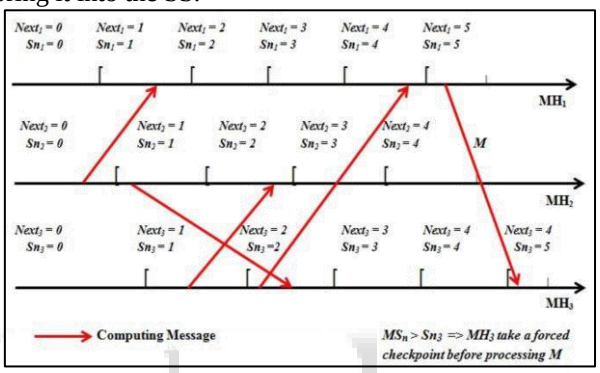

Fig. 5: Spontaneous checkpoints and forced checkpoints

For this purpose, the initiator process i send first the request to all its direct neighbors, and so on until its propagation in all the nodes of the network wherein no partition is allowed. At the reception of the request for the first time (the last checkpoint taken by the node i has its Sni lower than the received one), a node i takes a forced checkpoint and save it as a tentative checkpoint into the SS. We have a tentative checkpoint when a node takes it and saves it into the SS upon receiving a coordination request. On the other side, we call a permanent checkpoint, a tentative checkpoint that has been validated by the initiator once the coordination phase is achieved.

A real variable, noted weight (piggybacked in the requests and their responses), is set to 1 by the initiator (coordinator). Before sending a request, any node divides weight by 2 for each recipient. In the response, the remaining value of weight is sent. When the initiator receives the responses, it sums their weights. If the total is equal to 1, then the coordinator ends the coordination phase by converting its tentative checkpoint in permanent checkpoint on the SS and asks other nodes to do the same. At the end of the coordination phase, all the nodes delete their spontaneous and forced checkpoints saved locally whose Sn is less than the permanent checkpoint saved into SS. Moreover, if a node is already in phase 2, it must ignore all other coordination requests. In this case, it responds to the initiator of the new coordination request without propagating it to its own neighbors.

IV. ANALYSIS OF METHODS

Minimum process coordinated checkpointing does not consider useless checkpoints. Minimum number of processes takes the checkpoint. Energy consumption and recovery latency are reduced when a cluster head fails. It is difficult and time consuming to decide which process should take the checkpoints. Staggered checkpointing and recovery avoids simultaneous contention for resources and handles multiple failures. If the node fails, the data has to be searched and retrieved for recovery along with last saved checkpoint. Due to heavy message logging, the staggering should be discouraged in communication-intensive applications. Uncoordinated checkpointing protocol logs only a subset of the application messages and does not require restarting systematically all processes when a failure occurs. Uncoordinated checkpointing does not require any synchronization between the processes at checkpoint time. Thus, it can be used to address the problem of burst accesses to the I/O system by allowing to better schedule checkpoints. However uncoordinated checkpointing has a major drawback called the domino effect. Two phases algorithm for checkpointing supports both mobile and fixed stations. Spontaneous checkpoint limits data transfer to stable storage.

V. CONCLUSION

When designing an efficient ad hoc network application, we must consider the resource constraints and their scalability. Fault tolerance is a major research area in the MANETs. Consistent checkpointing is suitable for most of the long-running distributed applications. Lack of stable storage makes the task of checkpointing more challenging in MANET. Checkpointing algorithms for cluster based MANETs are developed for less overhead, reducing number of checkpoints for saving both time and memory space by using different approaches. We can use a better approach for node arrangement for checkpointing process or a hybrid checkpointing strategy can be used which is a combination of two or more checkpointing schemes.

REFERENCES

[1] Elnozahi, E.N., Alvisi, L., Wang, Y.M., Johnson, D.B.: A survey of rollback-recovery protocols in message-passing systems. ACM Computing Surveys 34(3), 375–408 (2002)
[2] S. Biswas and S. Neogy, Checkpointing and Recovery using Node Mobility among Clusters in Mobile Ad Hoc Network, Advances in Intelligent Systems and Computing, Vol.176, pp.447-456, 2012
[3] P. K. Jaggi and A. K. Singh, Staggered Checkpointing and Recovery in Cluster Based Mobile Ad Hoc Networks, Advances in Parallel Distributed Computing Communications in Computer and Information Science,Vol. 203, pp.122-134, 2011
[4] C. Men, Z. Xu and X. Li, An Efficient Checkpointing and Rollback Recovery Scheme for Cluster-based Multi-channel Ad-hoc Wireless Networks, Proceeding ISPA '08 Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications, pp.371-378, IEEE Computer Society Washington, DC, USA
[5] R. Tuli and P. Kumar, Asynchronous Checkpointing and Optimistic Message Logging for Mobile Ad Hoc Networks, International Journal of Advanced Computer Science and Applications, Vol. 2, No.10, pp. 70 – 76,2011
[6] K. Saluja and Praveen Kumar. Transitive Dependencies Tracking in Minimum-Process Checkpointing Protocol for Mobile Ad hoc Networks, International Journal of Computing Science and Communication Technologies, Vo. 4, No. 1, 2011 (ISSN 0974-3375)
[7] T.Y.T. Juang and M. C. Liu, An Efficient Asynchronous Recovery Algorithm In Wireless Mobile Ad Hoc Networks, Journal of Internet Technology, Vol. 3, No. 2, pp.143-152, 2002
[8] P.K. Jaggi and A.K.Singh, Message efficient global snapshot recording using a self-stabilizing spanning tree in a MANET, International Journal of Communication Networks and Information Security (IJCNIS), Vol.3,No. 3, pp.247-255, 2011

