Asynchronous Checkpointing and Optimistic Message Logging for Mobile Ad Hoc Networks

Ruchi Tuli
Affiliation 1 : Research Scholar, Department of Computer Science
Singhania University, Pacheri Bari (Rajasthan) INDIA
Affiliation 2**:** Department of Computer Sc. & Engg., P.O Box 31387, Yanbu University College,
Kingdom of Saudi Arabia

Parveen Kumar
Professor,
Department of Computer Science,
Meerut Institute of Engineering & Technology, Meerut
INDIA

***Abstract* - In recent years the advancements in wireless communication technology and mobile computing fueled a steady increase in both number and types of applications for wireless networks. Wireless networks can roughly be classified into cellular networks which use dedicated infrastructure (like base stations) and ad hoc networks without infrastructure. A Mobile Ad Hoc Network (MANET) is a collection of mobile nodes that can communicate with each other using Multihop wireless links without using any fixed infrastructure and centralized controller. Since this type of networks exhibits a dynamic topology, that is, the nodes move very frequently, it is hard to establish some intermittent connectivity in this scenario. Fault tolerance is one of the key issues for MANETs. In a cluster federation, clusters are gathered to provide huge computing power. Clustering methods allow fast connection and also better routing and topology management of mobile ad hoc networks (MANET). To work efficiently on such systems, networks characteristics have to be taken into account, for e.g. the latency between two nodes of different clusters is much higher than the latency between two nodes of the same cluster. In this paper, we present a message logging protocol well-suited to provide fault tolerance for cluster federations in mobile ad hoc networks. The proposed scheme is based on optimistic message logging.**

Keywords – MANETs; clusterhead; checkpointing; pessimistic logging; fault tolerance; Mobile Host.

I. INTRODUCTION

Wireless networks include infrastructure-based networks and ad hoc networks. Most wireless infrastructure-based networks are established by a one hop radio connection to a wired network. On the other hand, mobile ad hoc networks are decentralized networks that develop through self-organization [1]. The original idea of MANET started out in the early 1970s. At this time they were known as packet radio networks. Lately, substantial progress has been made in technologies like microelectronics, wireless signal processing, distributed computing and VLSI (Very Large Scale Integration) circuit design and manufacturing [2]. This has given the possibility to put together node and network devices in order to create wireless communications with ad hoc capability.

MANETs are formed by a group of nodes that can transmit and receive data and also relay data among themselves. Communication between nodes is made over wireless links. A pair of nodes can establish a wireless link among themselves only if they are within transmission range of each other. An important feature of ad hoc networks is that routes between two hosts may consist of hops through other hosts in the network [3]. When a sender node wants to communicate with a receiver node, it may happen that they are not within communication range of each other. However, they might have the chance to communicate if other hosts that lie in-between are willing to forward packets for them. This characteristic of MANET is known as multihopping. An example is shown in figure 1. Node A can communicate directly (single-hop) with node B, node C and node D. If A wants to communicate with node E, node C must serve as an intermediate node for communication between them. Therefore, the communication between nodes A and E is multi-hop.

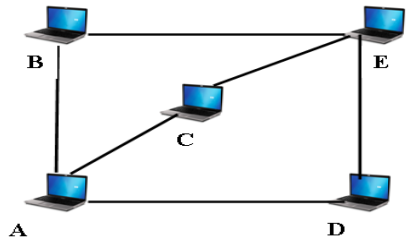

Figure 1 – Multi-hop communication in a mobile ad hoc network

Today wireless bluetooth, personal area networks (PAN), IEEE 802.11 a/b/g, wireless local area networks (WLAN) and HIPERLAN/2, are communication standards that include ad hoc features [4]. Figure 2 shows an example of a mobile ad hoc network composed of commonly used wireless devices.

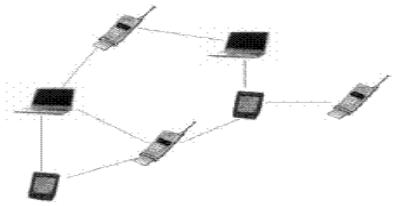

Figure 2 – Wireless Mobile Ad hoc Network

Checkpoint and message logging protocols are designed for saving the execution state of mobile application, so that when a MH recovers from a failure, the mobile application can roll back to the last saved consistent state, and restart execution with recovery guarantees. The existing protocols assume that the MH's disk storage is not stable and thus checkpoint and log information are stored at the base stations [5], [6].

Log-based rollback recovery exploits the fact that a process execution can be modeled as a sequence of deterministic state intervals, each starting with the execution of a non-deterministic event. A non-deterministic event can be the receipt of a message from another process or an event internal to the process. A message send event is not a non-deterministic event. Log-based rollback recovery assumes that all non-deterministic events can be identified and their corresponding determinants can be logged into the stable storage.

During failure-free operation, each process logs the determinants of all non-deterministic events that it observes onto the stable storage. Additionally, each process also takes checkpoints to reduce the extent of rollback during recovery. After a failure occurs, the failed processes recover by using the checkpoints and logged determinants to replay the corresponding non-deterministic events precisely as they occurred during the pre-failure execution. Because execution within each deterministic interval depends only on the sequence of non-deterministic events that preceded the interval's beginning, the pre-failure execution of a failed process can be reconstructed during recovery up to the first non-deterministic event whose determinant is not logged. The deterministic intervals composing the process execution are called state intervals. The state intervals are partially ordered by the Lamport's happen-before relation [7].

Message logging techniques are classified into pessimistic [8], optimistic [9], [10], [11] and causal [12], [13], [14], [15]. Pessimistic logging protocols assume that a failure can occur after any non-deterministic event in the computation. This assumption is "pessimistic" since in reality failures are rare. In their most straightforward form, pessimistic protocols log to the stable storage the determinant of each non-deterministic event before the event affects the computation. A pessimistic protocol is one in which each process p never sends a message until it knows that all messages delivered before sending the previously sent messages are logged. Pessimistic protocols will never create any inconsistent process (orphans), and so the reconstruction of the state of a crashed process is very straightforward. The pessimistic protocols potentially block a process for each message it receives.

In optimistic logging protocols, processes log determinants asynchronously to the stable storage. These protocols optimistically assume that logging will be complete before a failure occurs. Determinants are kept in a volatile log, and are periodically flushed to the stable storage. Thus, optimistic logging does not require the application to block waiting for the determinants to be written to the stable storage, and therefore incurs much less overhead during failure-free execution. However, the price paid is more complicated recovery, garbage collection, and slower output commit. If a process fails, the determinants in its volatile log are lost, and the state intervals that were started by the non-deterministic events corresponding to these determinants cannot be recovered. Furthermore, if the failed process sent a message during any of the state intervals that cannot be recovered, the receiver of the message becomes an orphan process and must roll back to undo the effects of receiving the message. Optimistic logging protocols do not implement the always-no-orphans condition.

Causal logging combines the advantages of both pessimistic and optimistic logging at the expense of a more complex recovery protocol. Like optimistic logging, it does not require synchronous access to the stable storage except during output commit. Like pessimistic logging, it allows each process to commit output independently and never creates orphans, thus isolating processes from the effects of failures at other processes. Moreover, causal logging limits the rollback of any failed process to the most recent checkpoint on the stable storage, thus minimizing the storage overhead and the amount of lost work. Causal logging protocols make sure that the always-no-orphans property holds by ensuring that the determinant of each non-deterministic event that causally precedes the state of a process is either stable or it is available locally to that process.

In this paper we focus on optimistic based message logging for communications in a clustered ad hoc network, since the checkpointing-only schemes are not suitable for the mobile environment and also in ad hoc environments in which unreliable mobile hosts and fragile network connection may hinder any kind of coordination for checkpointing and recovery. In order to cope with the storage problem, the task of logging is assigned to the CH instead of MHs, since each message heading to a MH is routed through the CH. Also, in order to reduce the overhead imposed on mobile hosts, cluster heads take charge of logging and dependency tracking, and mobile hosts maintain only a small amount of information for mobility tracking.

The rest of this paper is organized as follows: Section 2 discusses related work and problem formulation. In section 3, system model is described. Section 4 explains the basic algorithm and comparison with existing schemes is described in section 5. Finally section 6 concludes the paper.

II. Related work and Problem Formulation

A. Related Work

Application failure recovery in the mobile computing environment has received considerable attention in the recent years. The schemes that have been proposed employ checkpointing, logging or a combination of both, recognizing the inherent limitations of the mobile computing environments. Since the requirements of a ad hoc network are different from the mobile computing environment, these issues needs to be addressed in this area as well.

Prakash and Singhal describe in [16] a checkpointing algorithm for Mobile Conputing System. Checkpoint collection is synchronous and non-blocking. A minimum number of nodes are forced to take checkpoints. Each MH maintains a dependence vector. MHs maintain causal relationships through message. This scheme reduces energy consumption by

powering down individual components during periods of low activity.

In [17] T.Park et.al has presented an efficient movement based recovery scheme. This scheme is a combination of message logging and independent checkpointing. Main feature of this algorithm is that a host carrying its information to the nearby MSS can recover instantly in case of a failure. To enhance failure-free execution, concept of a 'certain range' is introduced. An MH moving inside a range , recovery information remains in host MSS otherwise it moves recovery information to nearby MSS. Though recovery is ensured, failure-free execution cost increases. Due to this out of range concept overheads due to transfer of checkpoint from one MSS to another MSS increases many fold.

Sapna E. George [18]et.al describes a checkpointing and logging scheme based on mobility of MHs. A checkpoint is saved when hand-off count exceeds a predefined optimum threshold. Optimum threshold is decided as a function of MH's mobility rate, failure rate and log arrival rate. Recovery probability is calculated and recovery cost is minimized in this scheme.

Acharya et al. [6] describes uncoordinated checkpointing, where multiple MHs can arrive at a global consistent checkpoint without coordination messages. However, neither it takes into account how failure recovery is achieved nor does it address the issue of recovery information management in the face of MH movement.

In [19] authors proposed an independent checkpointing scheme which saves the state of processes in the computer to which a mobile host is currently attached.

The authors in [20] presents a low overhead recovery scheme based on a communication induced checkpointing, which allows the processes to take checkpoints asynchronously and uses communication-induced checkpoint coordination for the progression of the recovery line. The scheme also uses selective pessimistic message logging at the receiver to recover the lost messages. However, the recovery scheme can handle only a single failure at a time.

P. Kumar and A. Khunteta [22] proposed a minimum-process coordinated checkpointing algorithm for deterministic mobile distributed systems, where no useless checkpoints are taken, no blocking of processes takes place, and anti-messages of very few messages are logged during checkpointing. In their algorithm they have tried to reduce the loss of checkpointing effort when any process fails to take its checkpoint in coordination with others.

B. Problem Formulation

Cluster federations are hierarchical systems. The latency between two clusters is much higher than the latency between two nodes of the same cluster. For efficient execution on such systems, applications must take into account the topology of the cluster federation. Communications between nodes of the same cluster should be favored over communications between nodes of different clusters.

The objective of the present work is to design an optimistic based message logging for communications in a clustered ad hoc network, since the checkpointing-only schemes are not suitable for the mobile environment and also in ad hoc environments. In order to cope with the storage problem, the task of logging is assigned to the CH instead of MHs, since each message heading to a MH is routed through the CH. In order to reduce the size of dependency information carried in each message for asynchronous recovery, only the messages between the CHs carry the information, and the dependency between the MHs residing in the same Cluster can be traced through the message order within the CH. Using the restricted dependency tracking, no extra overhead is imposed on MHs. Of course, there is a possibility of unnecessary rollbacks due to the imprecise dependency information, however, comparing with the checkpointing-only schemes, the chance of rollback propagation in the message logging schemes is very low.

III. System Model

A successful approach for dealing with the maintenance of mobile ad hoc networks is by partitioning the network into clusters. In this way the network becomes more manageable. Clustering is a method which aggregates nodes into groups. These groups are contained by the network and they are known as clusters. Clusters are analogous to cells in a cellular network. However, the cluster organization of an ad hoc network cannot be achieved offline as in fixed networks [21]. In most clustering techniques nodes are selected to play different roles according to a certain criteria. In general, three types of nodes are defined:

Ordinary nodes :- Ordinary nodes are members of a cluster which do not have neighbors belonging to a different cluster.

Gateway nodes:- Gateway nodes are nodes in a non-clusterhead state located at the periphery of a cluster. These types of nodes are called gateways because they are able to listen to transmissions from another node which is in a different cluster. To accomplish this, a gateway node must have at least one neighbor that is a member of another cluster.

Clusterheads:- Most clustering approaches for mobile ad hoc networks select a subset of nodes in order to form a network backbone that supports control functions. A set of the selected nodes are called clusterheads and each node in the network is associated with one. Clusterheads are connected with one another directly or through gateway nodes. The union of gateway nodes and clusterheads form a connected backbone. This connected backbone helps simplify functions such as channel access, bandwidth allocation, routing power control and virtual-circuit support.

Clusterheads are analogous to the base station concept in current cellular systems. They act as local coordinators in resolving channel scheduling and performing power control. However, the difference of a clusterhead from a conventional base station resides in the fact that a clusterhead does not have special hardware, it is selected among the set of stations and it presents a dynamic and mobile behavior. Since clusterheads must perform extra work with respect to ordinary nodes they can easily become a single point of failure within a cluster. For this reason, the clusterhead election process should consider for the clusterhead role, those nodes with a higher degree of

relative stability. The main task of a clusterhead is to calculate the routes for long-distance messages and to forward inter-cluster packets. Figure 3 shows the system model and different roles of nodes in a mobile ad hoc network organized by clusters.

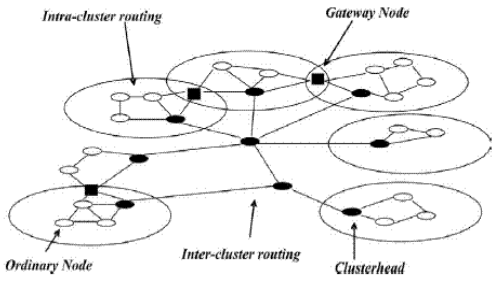

Figure: - 3 System Model

The clustering system considered in this paper follows the model presented in the figure above The system is organized into various clusters, each having a clusterhead and ordinary nodes, which will be termed as Mobile Hosts; a set of dynamic links can be established between a MH and a Cluster head. The area covered by a cluster head is called a *cell*. A MH residing in a cluster can be connected to the clusterhead servicing the cluster and the MH can communicate to another MH only through the cluster head. The links in the dynamic network support FIFO communication in both directions.

For a MH to leave a cluster and enter into another cluster, it first has to end its current connection by sending a *leave*(x) message to the cluster head, where x is the sequence number of the last message received from the cluster head, and then establish a new connection by sending *join* (MH-id, previous cluster-id, previous clusterhead-id) message to the new cluster head in the new cluster. Each cluster head maintains a list of identifiers, called a *Current_Nodelist*, with which nodes it connected at current time

A MH can also disconnect itself from the cluster head voluntarily without leaving the cluster by sending a *disconnect (x)* message to conserve power. When the cluster head receives a *disconnect* message from a Node, it marks that node to be "disconnected" by setting a flag that maintains a list of voluntarily disconnected MHs, called *disconnected_Nodelist* Later on, the MH can *reconnect* to any cluster by sending a *reconnect* (MH-id, previous cluster-id, previous clusterhead_id) message to the current cluster head. If the MH is reconnected to a new cluster, the new cluster head informs the previous cluster head of the reconnection so that the previous cluster head can perform the proper hand-off procedures.

IV. Proposed Algorithm

The ordinary Nodes in the clustered ad hoc networks are considered highly vulnerable to failures, while the cluster heads are relatively reliable as they are chosen among the nodes with a higher degree of relative stability. By reliable CH, we mean that recovery information for MHs can never be lost due to its own failure. With this assumption, the volatile memory space of a CH can be utilized as a stable storage to save checkpoints and message logs of MHs.

A. Data Structures and Notations

Chk_i^x = checkpoint sequence number
i=0……..n
j=0……..n
MH_i = No. of Mobile hosts
i=0……..n
CH = Cluster head
m_i^{rcv} =No. of messages a mobile host has received
i=0……..n
rcv=0……..n
$Locate_i$ =variable to retrieve information after failure of MH
chk_seq=sequence number of latest checkpoint
cp_loc=current ID of a cluster
cp_ch=current cluster head ID
msg_seq =sequence number of the first message logged after checkpoint
log_set=IDs of cluster heads that store MH logs
$timeToCkp_i$=Timer to take checkpoint on MH

B. Checkpointing and Message Logging

Each mobile host MH independently takes a checkpoint and a unique sequence number is assigned to each checkpoint. For the checkpointing, MH first saves its current state as a checkpoint and then transfers the checkpoint to CH to which it is currently connected. Chk_i^x denotes the xth checkpoint of MH_i. Each checkpoint is identified by a pair of (i, x). MH_i then sends the checkpoint to its current CH, say CH_p. Each Mobile host also maintains a variable m_i^{rcv}, to count the number of messages a mobile host has received, and the value of m_i^{rcv} is sent with the checkpoint to the cluster head. On the receipt of a checkpoint and other related information CH saves it into stable storage. The value of m_i^{rcv} sent with the checkpoint to CH is used to decide the correct position of the checkpoint with respect to logged messages.

Each CH_p also maintains the message log for the nodes residing in that cluster. Since each message that is delivered to MH in the cluster is routed through CH, logging of messages incurs little overhead. Let Msg_i^x denotes xth message delivered to MH_i . In addition, CH_p also logs the messages related to the mobility of the nodes, such as join, leave, disconnect and reconnect. Any of these messages sent from MH must also carry the value of m_i^{rcv} and sequence number is logged with message.

A node after a failure should be able to locate its latest checkpoint and the message log for recovery. Each mobile host also contains a variable $Locate_i$, to retrieve the information after failure. $Locate_i$ contains *chk_seq*, *cp_loc*, *cp_ch*, *msg_seq* and *log_set*. *chk_seq* stores the sequence number of latest checkpoint and *cp_loc* stores the ID of the cluster and *cp_ch* stores the ID of the CH that has recorded the latest checkpoint. Let this cluster be called $cluster_{in}$ and CH be called CH*p*. *msg_seq* denotes the sequence number for the first message logged after the checkpoint and *log_set* contains the IDs of the cluster heads that stores its logs. At every checkpoint, *cp_loc* is updated with the current CH, *cp_seq* is updated with the

sequence number of the latest checkpoint and *log_set* is cleared. At every logging activity, the IDs of the current CHs are added to *log_set* if it is not present already. $Locate_i$ is logged by the CHs which a MH has visited. When a mobile host joins or reconnects to a new cluster, say CH_p, it sends $Locate_i$ with the connection message. Also when mobile host disconnects itself from or leaves CH_p, it sends $Locate_i$ with disconnection message if information in the $Locate_i$ has been changed since the connection was established. CH_p on receipt of $Locate_i$ logs it with the message. Each mobile host also maintains a variable $timeToCkp_i$ which defines Time interval until next checkpoint

C. Algorithm

We describe pseudocode for the checkpointing and message logging protocol here

D. Checkpointing at MH

If ($timeToCkp_i$ = Expire) then
    $chk\_seq_i=chk\_seq_i+1$; *//increment checkpoint sequence number*
    Perform checkpointing , $Chk_i^{chk\_seqi}$
    Save (i, chk_seq_i, m_i^{rcv},) with $Chk_i^{chk\_seqi}$
    // updating the Locate field

    $Locate_i.chk\_seq=chk\_seq_i$;
    $Locate_i.cp\_loc= cluster_{in}$;
    $Locate_i.cp\_ch=p$;
    $Locate_i.msg\_seq= m_i^{rcv}+1$;
    $Locate.log\_set$=NULL;
    Send ($Chk_i^{chk\_seqi}$[i, chk_seq, m_i^{rcv}]) to CH_p
Else
    Continue computation;
If (CH_p=recv $Chk_i^{chk\_seqi}$) from MH
    Save ($Chk_i^{chk\_seqi}$[i, chk_seq, m_i^{rcv},]);
Else
    Continue computation

E. Message logging at CH

a) When cluster head delivers a message M, to Mobile host

    $msg\_seq_i=msg\_seq_i+1$;
    Insert ($M_i^{msg\_seqi}$ [i, msg_seq_i]) into Log;

b) When Mobile host receives a message from cluster head (CHp)

    If(p $\notin Locate_i.log\_set$)
    $Locate_i.log\_set=Locate_i.log\_set \cup$ p;

c) When Mobile host sends a message to Cluster head

    If (M $\in$ {*join, leave, disconnect, reconnect*})
        Send (M [$Locate_i$, $m_i^{r\_seq}$])

d) When Cluster head receives a message M, from mobile host

    If (M $\in$ {*join, leave, disconnect, reconnect*})
        Insert (M ($Locate_i$, m_i^{rcv})) into log;

F. Proof of Correctness

Theorem I :- If a MH fails, its state can be reconstructed independently

Proof :- Let MH_i state be [s_i^0, s_i^1, s_i^2….s_i^l) before failure, which indicates messages e_i^o, e_i^{x-1}, e_i^x, ….e_i^y, where $1 \le y$, e_i^x is the first message from the last checkpoint and e_i^y is the last message before failure. Since all the messages delivered to MH_i are logged in CH and $Locate_i.log\_set$ indicates the order in which MH_i has contacted CH since its last checkpoint. After a failure MH_i should rollback to the latest checkpoint and the logged messages in the same order and it can reconstruct the same state intervals as the ones before failure. Because all the messages sent and received events are recorded, the MHi's state can be reconstructed.

V. Handling Failures and Disconnections

We distinguish here failures and disconnections. Failures can be categorized as – mobile host falls and is damaged, lost or stolen, battery is discharged. Disconnections are termed as hand-off. Since the mobility rate of mobile hosts in ad hoc networks is very high, so while connected a mobile host can change its position and can join another cluster. This movement is termed as disconnection. We will discuss these two issues separately.

After a MH recovers from a failure, either a mobile host is in the same cluster or the cluster can be changed. When a MH_i recovers from a failure, it first sends a *join*(MH-id, previous cluster-id, previous clusterhead-id) to its current CH, say CH_p. CH_p checks it *Active_nodelist* list and *Disconnected_nodelist*. If MH_i is found in any of these lists that mean MH after a failure has recovered in current cell and thus the $Loacte_i$ must have been logged in CH_p.

Sometimes it may happen that CH is not able to find MHi either in *Active_nodelist* list or *Disconnected_nodelist*, which means that MH after a failure has moved to another cluster. In this case, firstly MH sends a *join* (MH-id, previous cluster-id, previous clusterhead-id) to its current CH, say CH_q. Now, CH_q will broadcast the recovery message, so that previous CH which has been contacted by MH_i can deliver the most recent $Loacte_i$ to CH_q. After CH_q receives the most recent $Locate_i$, it starts with the recovery procedure. During the recovery of MH_i, new messages heading to MH_i can be logged at CH_q. However, these messages are delivered to MH_i after consuming all the messages in the log. Only the MHs which have failed rollback to the latest checkpoint and replay the logged messages to ensure the global recovery line and no other MHs need to rollback.

A node after a disconnection should be able to locate its latest checkpoint and the message log for recovery. Each mobile host contains a variable $Locate_i$, to retrieve the information after failure which we have discussed in section 4.1. For each hand-off or disconnection, a MH within a cluster transfers the checkpoint and message logs to the current cluster head, so that the recovery information can be retrieved later from the cluster head. For a MH, MH_i, connected to the cluster head CH_p, in cluster CL_i first saves its checkpoints and message logs and updates the information in $Locate_i$. Let

$MH\_data_{(i,p)}$ denotes the checkpoints and message logs of MH_i saved by CH_p of CL_i. When MH_i leaves CL_i and joins another cluster head say CH_{new} of cluster CL_k, a hand-off procedure is initiated by CH_{new} sending a handoff-request for MH_i to CH_p. While the hand-off procedure is performed, the recovery information is transferred from CH_p to CH_{new}. Figure 4 depicts the sequence of events that take place for the recovery information transfer.

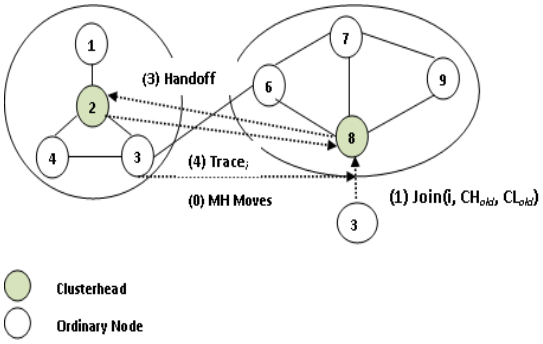

Figure 4 : Handling failures and disconnections

VI. Performance Comparison

In [1] authors proposed a communication pattern based checkpointing scheme to save consistent global states, in which a checkpoint is taken whenever a message reception is preceded by a message transmission. An independent checkpointing scheme which saves the state of processes in the computer to which a mobile host is currently attached was proposed by authors in [12]. Neither of the above approaches needs the checkpointing coordination, however, they may enforce a large number of checkpoints. [13] proposed a low-cost synchronous checkpointing scheme, in which a process can advance its checkpoint asynchronously, however, it may result in considerable message overhead and an inconsistency. [10] presents a low overhead recovery scheme based on a communication induced checkpointing, which allows the processes to take checkpoints asynchronously and uses communication-induced checkpoint coordination for the progression of the recovery line. The scheme also uses selective pessimistic message logging at the receiver to recover the lost messages. However, the recovery scheme can handle only a single failure at a time. In [18] authors describe a checkpointing and logging scheme based on mobility of MHs. A checkpoint is saved when hand-off count exceeds a predefined optimum threshold. Optimum threshold is decided as a function of MH's mobility rate, failure rate and log arrival rate. Recovery probability is calculated and recovery cost is minimized in this scheme.

We have described an optimistic based message logging scheme since checkpointing-only schemes are not suitable for ad hoc environments and most of the schemes described above are based on checkpointing-only approach. Also, we have followed the asynchronous checkpointing approach, as asynchronous recovery is desirable in ad hoc environments in which MH can be disconnected any time from the network and co-ordination may not be possible.

VII. Conclusion

In this paper, we have proposed an optimistic based message logging approach for cluster based ad hoc networks in which each MH in the cluster takes checkpoint independently. Also, each message that is delivered to MH in the cluster is routed through CH which avoids the overhead of message logging at MH. MH only carries minimum information and all the dependency tracking and mobility of MH can be properly traced by CH. The asynchronous checkpointing scheme relives the MH from any kind of coordination and they can take their checkpoints whenever they want.

References

[1] C. Prehofer, C. Bettstetter. “Self organization in communication networks: Principles and design paradigms”. IEEE *Communications Magazine*. Vol. 43. Issue 7. 2005. pp. 78-85.

[2] Y. P. Chen, A. L. Liestman, J. Liu. Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing. Clustering Algorithms for Ad hoc Wireless Networks. Vol. 2. Chapter 7: Nova Science Publishers, Hauppage NY, 2004. pp. 145-164. 7. J. Wu, J. Cao. “Connected k-hop clustering in ad hoc networks”. ICPP. 2005. pp 373-380.

[3] I. Chatzigiannakis, S. Nikoletseas. “Design and analysis of an efficient communication strategy for hierarchical and highly changing ad-hoc mobile networks”. *Mobile Networks and Applications*. Vol. 9. 2004. pp. 248-263.

[4] M. Frodigh, P. Johansson, P. Larsson. “Wireless Ad Hoc Networking--The Art of Networking without a Network”. *Ericsson Review*. Vol. 77. 2000. pp. 248-263.

[5] S. Gadiraju, Vijay Kumar, “Recovery in the mobile wireless environments using mobile agents”, IEEE Transactions on Mobile Computing, June 2004, Vol. 3.

[6] A. Acharya , B. R. Badrinath, “Checkpointing distributed applications on mobile computers”, in Proc. 3rd Int. Conf. Parallel and Distributed Information Systems, Austin, Texas, 1994, pp. 73–80.

[7] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM, 21(7):558–565, 1978.

[8] D. B. Johnson and W. Zwaenpoel, “Sender-Based Message Logging,” In Digest of Papers:17th International Symposium on Fault-Tolerant Computing, pp. 14-19, 1987.

[9] O. P. Damani and V. K. Garg, “How to Recover Efficiently and Asynchronously when Optimism Fails,” In Proc. the 16th International Conference on Distributed Computing Systems, pp. 108-115, 1996.

[10] D. B. Johnson and W. Zwaenepoel, “Recovery in distributed systems using optimistic message logging and checkpointing,” In Proc. the 7th Annual ACM Symposium on Principles of Distributed Computing, pp. 171-181, 1988.

[11] R. B. Strom and S. Yemeni, “Optimistic recovery in distributed systems,” ACM Transactions on Computer Systems, Vol.3, No.3, pp. 204-226, 1985.

[12] L. Alvisi, B. Hoppe, and K. Marzullo, “Nonblocking and Orphan-Free Message Logging Protocols,” In Proc. the 23th Symposium on Fault-Tolerant Computing, pp. 145-154, 1993.

[13] L. Alvisi and K. Marzullo, “Message Logging: Pessimistic, Optimistic, Causal and Optimal,”IEEE Transactions on Software Engineering, Vol.24, No.2, pp. 149-159, 1998.

[14] E.Elnozahy, “On the relevance of Communication Costs of Rollback Recovery Protocols,” In Proc. the 15th ACM Symposium on Principles of Distributed Computing, pp. 74-79, 1995.

[15] E. N. Elnozahy and W. Zwaenepoel, “Manet: Transparent rollback-recovery with low overhead, limited rollback and fast output commit,” IEEE Transactions on Computers, Vol.41, No.5, pp. 526-531, 1992.

[16] R.Prakash, M.Singhal, (1996) “Low Cost Checkpointing and Failure Recovery in Mobile Computing Systems”, IEEE Transacrions on Parallel and Distrinuted Systems, VOL. 7, NO. IO, OCTOBER 1996

[17] Taesoon Park, Namyoon Woo, Heon Y. Yeom,(2003) "An Efficient recovery scheme for fault tolerant mobile computing systems", Future Generation Computer System, 19(1): 37-53

[18] Sapna E. George,Ing-Ray Chen,Ying Jin, (2006)"Movement-Based Checkpointing and Logging for Recovery in Mobile Computing Systems", MobiDE, 51-58

[19] D.K. Pradhan, P. Krishna, and N.H. Vaiday. Recoverable mobile environment : Design and trade-off analysis. In Proc. of the 26th Int'l Symp. on Fault Tolerant Computing, 1996.

[20] D. Manivannan and M. Singhal. Failure recovery based on quasi-synchronous checkpointing in mobile computing systems. OSU-CISRC-796-TR36, Dept. of Computer and Information Science, The Ohio State University, 1996.

[21] A. B. McDonald, T. F. Znati. "A mobility-based framework for adaptive clustering in wireless Ad Hoc networks". IEEE *Journal on Selected Areas in Communications*. Vol. 17. 1999. pp. 1466-1487.

[22] Parveen Kumar, Ajay Khunteta, "Anti-message Logging based coordinated checkpointing protocol for Deterministic Mobile Computing Systems", International Journal of Computer Applications (0975-887), Vol. 3-No. 1, June, 2010.

[23] Parveen Kumar, "A Low-Cost Hybrid Coordinated Checkpointing Protocol for Mobile Distributed Systems", Mobile Information Systems [An International Journal from IOS Press, Netherlands] pp 13-32, Vol. 4, No. 1, 2007. [Listed in ACM Portal & Science Citation Index Expanded]

[24] Lalit Kumar, Parveen Kumar "A Synchronous Checkpointing Protocol for Mobile Distributed Systems: A Probabilistic Approach", International Journal of Information and Computer Security [], pp 298-314, Vol. 3 No. 1, 2007. [An International Journal from Inderscience Publishers, USA, Listed in ACM Portal]

[25] Sunil Kumar, R K Chauhan, Parveen Kumar, "A Minimum-process Coordinated Checkpointing Protocol for Mobile Computing Systems", International Journal of Foundations of Computer science,Vol 19, No. 4, pp 1015-1038 (2008). [Listed in Science Citation Index Expanded]

[26] Parveen Kumar, Lalit Kumar, R K Chauhan, "A Hybrid Coordinated Checkpointing Protocol for Mobile Computing Systems", IETE journal of research, Vol. 52, Nos 2&3, pp 247-254, 2006. [Listed in Science Citation Index Expanded]

[27] Lalit Kumar, Parveen Kumar, R.K. Chauhan, "Logging based Coordinated Checkpointing in Mobile Distributed Computing Systems", IETE Journal of Research, vol. 51, no. 6, pp. 485-490, 2005. [Listed in Science Citation Index Expanded]

[28] Obaida, M. A., Faisal, S. A., & Roy, T. K. (2011). AODV Robust (AODV R): An Analytic Approach to Shield Ad-hoc Networks from Black Holes. IJACSA - International Journal of Advanced Computer Science and Applications, 2(8), 97-102.

[29] Journal, I., Science, A. C., & Hod, M. (2011). A Survey on Attacks and Defense Metrics of Routing Mechanism in Mobile Ad hoc Networks. IJACSA - International Journal of Advanced Computer Science and Applications, 2(3), 7-12.

[30] Indukuri, R. K. R. (2011). Dominating Sets and Spanning Tree based Clustering Algorithms for Mobile Ad hoc Networks. IJACSA - International Journal of Advanced Computer Science and Applications, 2(2).