MOBILITY BASED CHECKPOINTING AND TRUST BASED RECOVERY IN MANET

Suparna Biswas[1], Sarmistha Neogy[2] and Priyanka Dey[3]

[1,3]Department of Computer Science & Engineering, West Bengal University of Technology, Salt Lake, Kolkata
mailtosuparna@gmail.com
priyankadey24@yahoo.co.in

[2]Department of Computer Science & Engineering
sarmisthaneogy@gmail.com

ABSTRACT

Proposed work is a mobility based checkpointing and trust based rollback recovery algorithm to provide fault tolerance in Mobile Ad hoc Network (MANET). Here each mobile host maintains a count of number of clusters a mobile host traverses through, during a single checkpoint interval. A mobile host increments 'cluster-change-count' by 1, each time it leaves a cluster and joins another. Each mobile host saves a checkpoint independently if its 'cluster-change-count' exceeds a predefined threshold. This measure is important because each mobile host leaves its last checkpoint and logs at different clusters that it has visited earlier. If the mobile host fails, the time to search and collect its last checkpoint and logs gets added to the recovery time of the mobile host. In MANET, retrieval of checkpoint and logs has to be done through a number of intermediate mobile hosts because each mobile host has short area coverage, hence direct communication among distant mobile hosts is not possible. Now if any of these mobile hosts fail, depending on the nature of failure, the checkpoint or log may be lost or forwarding of them to the failed host may be delayed causing unsuccessful or delayed recovery of the failed host respectively. This can be avoided if it is ensured that the checkpoints and logs are forwarded only through trusted nodes. Trust model proposed here computes trust value of a mobile host based on four factors: failure rate, availability in network, unused energy and recommendations from neighbour mobile hosts. Simulation results show that proposed algorithm achieves low recovery cost and high recovery probability of failed mobile hosts.

KEYWORDS

MANET, checkpoint, mobility, trust, recovery

1. INTRODUCTION

Rapid development of communication technology from wired to wireless network led almost all service oriented systems to "all time everywhere" service from "anywhere anytime" service. This has been possible due to stabilization of portable devices e.g. laptop, smart phones, mobile phones and advancement of wired communication to wireless infrastructure as well as wireless infrastructure less communication. Today's portable devices are equipped with sufficient resources hence can do computing as well as communication while on the move. In this kind of systems, computing devices as well as communication links are failure prone. Mobile hosts(MH) in MANET are not physically protected hence can come easily under intruder's control. Failure prone nodes are security attack prone and vice versa. In failure prone system fault tolerance measures should be there for quick recovery. Checkpoint and rollback recovery is popular technique for fault tolerance in mobile computing [1], [2], [3], [4]. In a checkpoint the state of a process saved contains process structure, register values, segments, actual data and open file information [5]. Starting address of each segment of a process to be checkpointed, size of a segment, actual data of a segment etc. are saved in checkpoint file. The recovering process will recover and resume execution from saved register values in checkpoint file. The checkpoint is generally saved in cluster head. Now the MH moves across different clusters and may fail in other cluster. To recover from saved checkpoint which is saved in another cluster head, the checkpoint needs to be transferred to the cluster head in which the failed host will recover. Checkpoint is transferred through a number of cluster heads and gateway nodes. If any of these nodes fail, then the checkpoint may be lost and the recovery process will be terminated. So, the challenge is to select a set of nodes through which checkpoint can be successfully transferred without any fail. In this paper we propose a mobility and trust based checkpointing algorithm. Here we consider cluster based hierarchical MANET [6]. A MH moves across different clusters, leaves behind checkpoint and logs in different cluster heads. If the MH fails, these dispersed recovery information need to be collected at the time of recovery. To limit this search and retrieval cost of recovery information, MHs save checkpoint based on threshold of cluster_change_count [7]. All the cluster members of a cluster save checkpoints in current cluster head and if any cluster member fails in another cluster, last saved checkpoint of failed host is transferred through intermediate gateway nodes and cluster heads to the cluster head in which the failed host will recover. So, cluster head and gateway nodes must be available in the network without any failure. We compute trust value of a mobile host based on following four factors : failure rate, availability of node, unused battery power and recommendation from neighbor node. A mobile host having low failure rate and high unused battery power, will be highly available in network ensuring successful execution of any assigned task enhancing positive recommendations from neighbor nodes. We define a threshold of trust value. A mobile host is considered as trusted if its trust value is greater than threshold.

The rest of the paper is structured as follows: Section 2 discusses some of the related works, our observation and problem definitions. Section 3 describes system model, preliminary assumptions, data structures and notations. Section 4 explains proposed checkpointing scheme and proposed trust model. Section 5 presents algorithms and its correctness proof. Section 6 illustrates simulation and performance analysis. In Section 7 we conclude our work.

2. RELATED WORKS

The proposed work combining trust model with checkpoint and rollback recovery technique to implement secure checkpointing is unique of its kind. Hence similar related work cannot be presented. Overviews of some of existing works related to our proposed algorithm are described in this section.

In [7] Biswas and Neogy present a movement based independent checkpointing and log based rollback recovery algorithm in cluster based MANET. Mobile hosts move among different clusters and leave saved checkpoint and logs scattered in different previously visited cluster heads. If number of inter-cluster movements of a mobile host exceeds a predefined threshold then the mobile host takes checkpoint. Cluster head is like any other node and does not have stable storage, hence copy of checkpoint of a mobile host is saved in one of the cluster members of same cluster having sufficient resources. Threshold of Cluster_change_count limits recovery cost. In [8] P.K.Jaggi et.al describes a staggering based synchronous checkpointing and recovery scheme adapted for handling the limited storage and bandwidth problems of

Mobile ad hoc network which is cluster based. Here cluster member will save checkpoint and logs in cluster head. Cluster member will save only in channel message in log. Any cluster head can initiate checkpointing scheme. After taking checkpoint cluster head sends a request message to take checkpoint to the other cluster head s in its transmission range and then initiates the checkpoint in its cluster. In [9], Juang and Liu presented an efficient asynchronous recovery algorithm in cluster based MANET. In this work they have studied the fundamental problem of crash recovery in mobile distributed environment. Each processor takes checkpoint independently without any synchronization and coordination overhead. Yi, Heo, CHo and Hong presented an adaptive mobile checkpointing facility for wireless sensor networks in [10]. Each node saves copy of checkpoint in its neighbor nodes. A node saves checkpoint, sends its copy to all its neighbor nodes which send acknowledgement along with a probability. This probability is inversely proportional to the number of neighbor nodes of a node. As a node cannot save all the checkpoints of its neighbor nodes due to limited storage space, this probability helps each node to adaptively decide on saving checkpoints of its neighbor nodes calculated based on each node's neighbor node. In [11] X. Zhao et.al proposed a work to provide security in Mobile ad hoc network based on availability based trust model. They divided availability into three main elements: collaboration, honesty and ability. They build their trust model based on security and safety capacity of node, experience of some certain trusted nodes and ability of some certain nodes which participate in web collaboration and stand alone capacity of data transfer. Their trust model based security mechanism ensures the validity of trust computation under limited resource. In [12] A.fathi et.al proposed an energy efficient topology control protocol for Mobile ad hoc network which ensures minimum connectivity in the network at all times. They defined energy is the main factor of life time of a node. So, dynamic topology depends on energy. So, based on energy cluster head and gateway is elected so that they get maximum lifetime. In [13] J. Luo et.al proposed an efficient trusted node discovery method on recommendation based on fuzzy logic. They decide based on recommendation whether a node is trusted or selfish. In [14] R. Manoharan et.al proposed a trust based Hybrid Gateway selection scheme. According to their proposed scheme gateway is selected based on three metric: load taking capacity, recommendation based on observation from other neighbour node and route selection based on trust.

2.1. Problem Identification

After analyzing the above literature survey we identify the following problems that motivated us to propose the work.

- **Cluster Head has limited memory space:** In [8], checkpoints are always saved in cluster head. A cluster head has limited memory space. Hence, there should be an upper limit of number of cluster members that can be connected to a single cluster head. But they do not specify number of Cluster member in a cluster so that cluster head can save all the cluster members' checkpoint.
- **Random mobility of clusterhead and cluster member nodes:** Above mentioned related works do not consider random mobility of cluster members and cluster heads as a factor of saving checkpoints.
- **Frequent and sudden failures of mobile hosts:** Frequent transient or crash failures may occur in mobile hosts suddenly. Existing works consider that all mobile hosts that participate in checkpoint- recovery process do not fail.

- **Trust value calculation based on a single factor:** In [11], [12], [13] cluster head is selected based on only a single factor. In [11], cluster head is selected based on a mobile host's availability. In [12], mobile node is selected on the basis of only available battery power. In [13], cluster head is selected based on only recommendation from others. The problem of considering a single factor is that other factors that are not considered at all, may cause failure.

In our work we have addressed above problems and tried to give solutions as given below:

- **Limited cluster members in a single cluster:** Maximum number of cluster member nodes of a cluster is defined by dividing allocated memory space of a cluster head to save copy of checkpoints of its cluster member nodes by size of checkpoint.
- **Inter-cluster movement based checkpointing:** Mobile hosts save checkpoints based on their inter-cluster movement.
- **Only trusted nodes evaluated based on multiple factors can participate in checkpoint-recovery process:** Trust value of a mobile host is calculated based on its failure rate, unused battery power, availability in the network and recommendation from others based on previous performance.

3. SYSTEM MODEL

MANET considered here consists of a number of clusters each having a cluster head and multiple cluster member nodes. Cluster heads communicate with each other through gateway nodes. In MANET there is no stable storage, fixed access point, topology changes dynamically.

Assumptions: Our proposed algorithm is true based on following assumptions:

- Recovery of a mobile host is possible for both transient and crash failures.
- Each mobile host saves last checkpoint, logs and log_tags of current checkpoint interval at any instant of time.
- The mobile host that saves checkpoint and the cluster head that carries the copy will not fail simultaneously

3.1. Data Structures & Notations:

CH = Cluster head, CM =cluster member, GW = gateway, c3 = cluster_change_count, c3_th = threshold of cluster change count, F_r = failure rate, $\%B_{left}$ = percentage of available battery power, R_{t_chkp}= Rate of checkpoint transfer, A = availability, R= Recommendation from other node, T_r = recovery time, T_{val} = trust value, T_1= Time required to send CH id where checkpoint and log is saved to the currently connected CH, T_2 = Time required to send request to the CHs where checkpoint and log reside, T_3= Time required to transfer checkpoint and log from requested CHs to the CH where failure CM is connected after failure,T_4= Time required to transfer checkpoint and log from currently connected CH to failed CM,T_5= Time required to rollback,T_6= Time required to replay log, mh_dep [ ] = during current interval, list of mhs from which computation messages received, CH_mh_list[ ] = list of mobile hosts connected to a clusterhead currently, CH_mh_list[i] = 1, mh_i connected, CH_mh_list[i] = 0, mh_i disconnected, CH_mh_list[i] = -1, mh_ifailed, CH_traversed_list[ ] = Array of cluster heads through which a mobile host traverses during a checkpoint interval, mhi_s = mobile host that sends computation message, mhi_r = mobile host that receives computation message, m_c= computation message, m_{co} = coordination message, log = $log_{s,\ r,intv,seq}$: s = sender mh, r = receiver mh, intv. = interval, seq = sequence number of computation message, log_tag = $log_{\mathbf{r,s,intv,seq}}$, log $_{\mathbf{(1,*,\ 1,\ *)}}$: s = $mh_{1,}$ intv = 1, r = seq = 'any', UID = UID $_{\mathbf{s,\ r,intv,seq}}$ **:** s=sender mh, r = receiver mh, intv. = interval, seq = sequence number of computation message, e.g. UID $_{(*,\ 1,\ 1,*)}$: r = mh_1, intv = 1, s = seq = 'any'

Checkpoint_ file:	**MH_id, CH_id** **Process status** **Data, intv**	**Log_file:**	**$log_{s,\ r,intv,seq,}$** **copy of m_c**

4. PROPOSED WORK

4.1 Checkopinting and Recovery Technique

During a checkpoint interval mobile host computes, sends and receives computation messages to and from other mobile hosts. A mobile host saves log of sent computation message and forwards to current cluster head. A cluster head saves logs of its cluster members, coordinates recovery of its failed member hosts besides own computations. A mobile host increments 'cluster-change-count' by 1 each time the mobile host leaves a cluster and joins another. Each mobile host saves a checkpoint independently if its 'cluster-change-count' exceeds a predefined threshold. If a mobile host fails before saving a checkpoint during current checkpoint interval, the cluster head in which it will recover will search for its last checkpoint and logs saved in other cluster heads. The cluster head that saved failed MH's last checkpoint will forward checkpoint to the cluster head in which the failed MH will recover. The cluster head then forwards checkpoint to the MH. In similar way the logs also will reach the MH. Now the MH will roll-back upto last checkpoint, replay logs, sends request message to the MHs from which the MH has received computations messages according to log_tags saved before failure. The recovered mobile host is now same as that before failure.

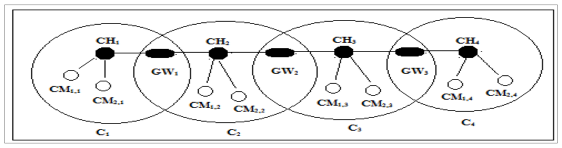


Figure.1. Working example of the network considered in our work

Above figure represents the cluster based mobile ad hoc network considered in our work. In the above example, we consider 4 clusters $C_{1,}$ $C_{2,}$ $C_{3,}$ C_4. Each cluster contains 4 cluster members. Within 4 cluster members, one cluster member will play the role of cluster head, another cluster member will play the role of gateway node, rest of the two are normal cluster members. So clusters C_1, C_2, C_3 and C_4 have cluster heads CH_1, CH_2, CH_3 and CH_4 respectively. Each cluster head of one cluster is connected to cluster head of another cluster through gateway nodes, GW_1, GW_2 and GW_3. Here we consider each cluster member will take checkpoint when c3_th =3. So, for example we consider $CM_{1,1}$ initially take checkpoint at C_1 and save checkpoint and logs in $CH_{1..}$So after visiting $C_{2,}$ C_3 when CM_1 will reach C_4 its c3_th will be 3. So CM_1 will have to save checkpoint and logs in CH_4. After taking checkpoint in C_3 two cases may happen: case1:$CM_{1,1}$ may fail before saving checkpoint CH_4, case2: $CM_{1,1}$ may save checkpoint and logs in CH_4.

Case1: If $CM_{1,1}$ fails before saving checkpoint to CH_4, after reconnect to CH_4 after failure $CM_{1,1}$ will give the id of CH_1 where checkpoint and logs of send messages [15] are saved. Then CH_4 will send requests to CH_1 through the path CH_4-GW_3-CH_3-GW_2-CH_2-GW_1-CH_1.CH_1will send checkpoint and logs through the above discussed path.CH_4 then sends checkpoint and logs to $CM_{1,1}$. $CM_{1,1}$ will rollback to this checkpoint and replay the logs. This mobility based checkpoint and log based rollback recovery is consistent. If mobile hosts save checkpoint independently and do not saved log of send message then if a mobile host fails without saving next checkpoint whereas the receiver host of that message has saved checkpoint then the message has been received without being sent i.e. orphan message which will cause inconsistent recovery. Here our proposed checkpointing and recovery algorithm is consistent.

Case2: $CM_{1,1}$ saves checkpoint and logs to CH_4 and sends message to CH_4 to delete last saved checkpoint and logs which is saved in CH_1. Ch_4 sends message to $CH_1$to delete checkpoint and logs of $CM_{1,1}$. After receiving message CH_1 deletes checkpoint and logs of $CM_{1,1}$ and acknowledges CH_1.

This recovery technique does not consider failure probability of intermediate MHs through which the checkpoint is transferred to the failed mobile host from the cluster head that saved it. So if any checkpoint forwarding nodes fails, checkpoint transfer will be delayed or unsuccessful depending on the type of failure. Any of these are undesirable. Alternatives may be: i) Cluster head will keep a copy of the checkpoint till a successful recovery message comes from failed MH after its recovery. Then the cluster head will delete the checkpoint. Within stipulated time if successful recovery message does not come or failed recovery message come, the cluster head will again resend a copy of checkpoint and wait for reply. Reasons of this could be due to failure of forwarding mobile hosts. In worst case, this may continue repeatedly causing resource consumptions of cluster head, intermediate forwarding mobile hosts.

ii) If it can be ensured that checkpoint forwarding is done through only 'trusted MHs' that does not fail during recovery process of a MH. To ensure that, before forwarding checkpoint to next MH, current MH must be sure about trust status of next MH. For that, trust value of next MH has to be calculated. That can be done using a trust model [6].

The second alternative solution is much more acceptable because checkpoint forwarding through trusted MHs will ensure that checkpoint will reach to the failed mobile host. Here we propose a trust model that evaluates trust of a mobile host based on multiple factors like its unused battery power, life time in the network, failure rate and recommendation from other MHs.

4.2 Proposed Trust Model

Trust of cluster member in a cluster is a belief that all the cluster member will resist different types of attack. A cluster member can be trusted or distrusted. But sometimes it may be impossible to decide whether a cluster member is 'trusted' or 'distrusted' due to lack of communication or improper knowledge. In this case we consider cluster member as 'uncertain' cluster member. According to [16], probability of a mobile host being trusted, distrusted and uncertain can be expressed in following way-

$$b + d + u = 1 \qquad \text{.................................} (1)$$

Where b=belief, d=disbelief and u=uncertain. In our paper we term belief as trusted, disbelief as distrusted.

There are various methods of trust value evaluation of a system [11, 13] and how to remove uncertainty [17]. In proposed algorithm trust value is defined based on trust matrix in which values of availability, recommendation from neighbour nodes, failure rate, and unused energy per mobile host are saved.

Availability: This is a statistical data accumulated during a MH's communications with other mobile hosts. Each MH estimate its neighbour's availability based on its accumulated observations using Bayesian inference as discussed in [17]. Bayesian inference is a statistical inference in which evidence or observations are used to update statistics. Beta distribution, Beta (a, b), are used here in the Bayesian inference, since it only needs two parameters that are continuously updated as communications are made. When a new communication is made, if it is successful, then parameter 'a' is updated, otherwise parameter 'b' is updated. Every mobile host always maintains experience statistics record of its neighbour mobile hosts. Every mobile host always maintains availability record of its neighbour mobile host. When availability of a node is being observed or calculated, availability record from its neighbour hosts are taken.

Algorithm to evaluate availability:

```
if ( a >= b && a != 0) availability = 1;
else {  if ( b= = 0 )   availability = 0.5;
        else  availability = 0; }
```

Failure rate: It is an important factor to decide whether a MH is trusted or distrusted. According to our algorithm, a mobile host having low failure rate will be considered as trusted because it will not fail frequently and hence will be recommended by others.

Unused energy: Energy is also an important factor to decide whether a MH is trusted or distrusted. If a MH has high energy then its probability of disconnection from network will be low. So, MH will be more reliable.

Recommendation from neighbour nodes: When we have to decide whether a thing is good or bad, sometimes it is necessary to take recommendation from others. Many trust models are already implemented based on this concept of recommendation. So, we also include this parameter in the trust matrix defined here. Recommendation from others is a very important factor to get to know about the behaviour of a particular MH. In MANET, as no dedicated router is there, forwarding of checkpoint has to be done through normal mobile hosts. To select next node on a path, recommendation helps. A mobile host will give recommendation about other host based on its availability, failure rate and unused energy.

For example, suppose A will have to give recommendation of B. In that case, first B has to share its failure rate, availability and unused battery power to A. Then B checks its availability by using algorithm as already discussed in the above availability section. Then give recommendation given as below:

Table 1. Set recommendation value on different factors

Recommendation	**Failure rate (F_r)**	**Availability(A)**	**Unused battery power(P)**
1	< 0.5	> 0.5	> 0.5
0.5	= 0.5	= 0.5	= 0.5
0	> 0.5	< 0.5	< 0.5

So, a mobile host having failure rate < 0.5, availability > 0.5 and unused battery power > 0.5 will have a recommendation 1 from other mobile hosts. Fifty-fifty recommendation is given for all parameter value of 0.5. Hence, 0.5 is threshold level of all three parameters.

So, Trust_value_threshold ($\mathbf{T_{val\_th}}$) = (F_{r_th} * r) + ($A_{\_th}$ * r) + ($P_{\_th}$ * r) ---------------------- (2)

= (0.5 * 0.5) + (0.5 * 0.5) + (0.5 * 0.5)

= 0.25 + 0.25 + 0.25 = 0.75

By using following algorithm we consider whether a node is trusted, distrusted or uncertain.

<u>How to decide a mobile host's trust status:</u>

```
if ( ( trust_value >= trust_value_threshold ) && A >=.5&&F<=.5&&P>=.5 )
 mobile host is 'trusted' ;
else { if ( A = = 0 ) mobile host is 'uncertain' ;
else mobile host is distrusted ; }
```

If a node is uncertain, we check failure rate of the node. If failure rate is greater than or equal to 0.5, mobile host will be considered as trusted, otherwise distrusted. For example, suppose the trust matrix of three mobile host of a cluster is as follows:

Table 2. An example of labelling of node after computing trust value

MH	**F_r**	**A**	**P**	**R**	**$T_{\_val}$**	**Label of node**
A	0.2	0.2	0.7	(1,0,1)	= (0.2*1) + (0.2*0) + (0.7*1) = 0.2+0+0.7 = 0.9 > 0 .75	Trusted
B	0.2	0	0.7	(1,0,1)	= (0.2*1) + (0*0) + (0.7*1) = 0.2+0+0.7 = 0.9 but A==0	Uncertain
C	0.6	0.02	0.6	(0,0,1)	= (0.6*0) + (0.02*0) + (0.6*1) = 0+0+0.6 = 0.6 < 0.75	distrusted

In the above table trust value of A is greater than threshold and availability is non-zero. So it is a trusted node. Trust value of node C is less than threshold, so distrusted. Trust value of B is greater than threshold but availability is zero. So, it is uncertain. Justification is that when a mobile host is not available in the network, how its trust status can be defined?

In a cluster, the mobile host that has highest trust value will be selected as cluster head, the mobile host with next higher trust value will be gate way node. A mobile host once trusted may not be trusted forever because its battery power may be reduced, its availability in the network may be poor, its failure rate may be high and if any or all of these are true, then other nodes may not recommend it. Then a trusted node may become distrusted or undefined. If due to reduced trust value, or random mobility or any other reason, cluster head or gateway nodes get changed, these nodes will transfer all the saved checkpoints, logs, data structures to the newly selected cluster head or gateway nodes respectively. The node may change but its ID will remain same like cluster head of cluster 1 is designated as CH_1. Now irrespective of the node, the cluster head ID of cluster 1 will always be CH_1.So trust value evaluation of mobile hosts is a continuous process which must repeat itself after certain time interval which can be set based on specific system or the network being used. As only trusted nodes can be cluster head and gateway nodes, by our proposed trust model it is ensured that checkpoint will be forwarded through only trusted nodes.

5. ALGORITHM

1. Mobile hosts compute, communicate with each other through message passing.
2. Each mobile host sends a beacon message to own cluster head at regular interval to convey that it is connected. Through beacon message continuous authentication of nodes also goes on.
3. If mh_{i_s}wants to send a computation message to mh_{i_r}, mh_{i_s}saves a log, forwards m_c and log to current cluster head. c_ch saves log in its memory, checks own ch_mh [ ], if finds a 1 in corresponding array field then $mh_{i_r}$is in cluster 1, if finds a 0 then mhi_r is in other clusters, if finds a -1, then mh_{i_r}failed.

   case 1: $mh_{i_s}$and mh_{i_r}in same cluster:

   c_ch forwards m_c to mh_{i_r}andmh$_{i_r}$sends ack to mh_{i_s} throughc_ch.

   case 2: $mh_{i_s}$and mh_{i_r}in different cluster:

   c_ch of mhi_s broadcasts a look up (mh_{i_r}) message to all cluster heads through gateway nodes. All cluster heads searches their corresponding ch_mh [ ]. The CH that founds 1 replies to c_chof mh_{i_s}, c_ch forwards m_c and UID of m_c gateway node which saves UID of m_c and forwards m_c to replyingch, ch forwards to mh_{i_r}.

   case3: mh_{i_r}fails:

   mh_{i_s} stops sending m_c temporarily, will try to send later.
4. mh_{i_r} receives computation message,saves updates dependency array, sends UID of m_c to gateway node.
5. mh_{i_s}moves randomly, leaves current cluster and joins another cluster .
6. In cluster based adhoc network each mobile host increments cluster_change_counter (c3) each time a mobile host changes cluster.
7. The mobile host saves ID of cluster heads being traversed in an array, CH_traversed_list[ ].
8. Before the mobile host leaves a cluster head, sends different data structures e.g. dependency array, CH_traversed_list etc. saved being connected to current cluster head along with a leave message to current cluster head.
9. The mobile host saves ID of previous cluster head in CH_traversed_list[ ], e.g. CH_traversed [0] = CH_1, if mh leaves CH_1.
10. The mobile host next joins another cluster head say CH_2
11. repeat steps 2 to 6 .
12. The mobile host saves ID of previous cluster head in CH_traversed_list[ ], e.g. CH_traversed [1] = CH2, if mh leaves CH2.
13. Repeat steps 2 to 6.
14. The mobile host saves ID of previous cluster head in CH_traversed_list[ ], e.g. CH_traversed [2] = CH3, if mh leaves CH3.
15. If c3 > c3_th, mobile host invokes checkpoint procedure ( ).

    15.1.The mh takes a snapshot of current state of computation of running application process.

    15.2 Checkpointingmh searches for an immediate neighbour with highest remaining memory and saves a back up copy of checkpoint in it. The mh ends checkpoint_backup_node = ID of$mh_{neighbour}$ to CH_2.
16. The mobile host sends delete_logof prev_interval message to CH_2 which forwards this message tothe cluster heads that are saved in CH_traversed_list of the mobile host through gateway nodes.

16.1 If any gateway node finds a match between log_ID and UID, sends do_not_delete message to the cluster head.

16.2 The cluster head forwards log to the gateway node.

16.3 Gateway node saves log.

17. The mobile host enters into next checkpoint interval.
18. The mobile host saves current cluster head in its CH_traversed_list[ ]
19. repeat steps 2 to 6 till c3 ≤ c3_th.
20. The mobile host fails, last checkpoint can be recovered from the host itself or cluster head

20.1 **CASE 1:** Last checkpoint can be recovered from the failed host itself, the host rollsback upto last checkpoint.

20.1.1 **Case 1:** The failed host is mhi_s, Replay_ log (UID) message to c_ch, c_ch checks if the log is saved in its memory, if yes replays log to mhi_r, else broadcasts replay_log (UID) message to the chs saved in failed host'sch_traversed_list.

20.1.2 **Case 2:** The failed host is mhi_r, Receive_m_c (UID) message to c_ch, c_ch checks if the log is saved in its, memory, mhi_r receives mc from, c_ch, else c_ch broadcasts receive_$m_{c,}$ (UID) message to the chs saved in mhi_s' ch_traversed_list.

20.1.3 **Case 3:** The failed host is both mhi_s and mhi_r, Do tasks in case 1 and case2.

20.2 **CASE 2:** Last checkpoint needs to be recovered from cluster head, Current cluster head sends recovery_message of the failed mobile host to the cluster head saved in first field of ch_traversed_list. Then that cluster head forwards copy of checkpoint of failed host saved to it following the same path through which the recovery_message has been sent to it.

21. The failed host rolls back upto last checkpoint and recovers following any of the above three cases.

5.1 Correctness Proof

Theorem 1: The algorithm ensures consistent recovery

Proof: With the help of following two lemmas the above statement can be proved.

Lemma 1: There is no orphan message.

Proof: save log of each sent $m_{c,}$ TRUE for ∀ mh
if (mh fails without saving checkpoint)
sent m_c is retrieved from saved logs, TRUE for ∀ sent m_c

Lemma 2: There is no lost message.

Proof: save UID of each received $m_{c,}$ TRUE for ∀ mh
if (mh fails without saving checkpoint)
received m_c is retrieved according to saved UIDs, TRUE for ∀ mh

Theorem 2: Proposed algorithm causes minimum checkpoint and log overhead per mobile host per checkpoint interval

Proof: Each mh saves log, ∀ sent computation message, interval=current
Each mh saves UID, ∀ received computation message, interval=current
Each mh saves checkpoint, C3>C3_th, delete logs and UIDs, interval=next
Each mh saves log, ∀ sent computation message, interval=current
Each mh saves UID, ∀ received computation message, interval=current

Each mh saves checkpoint, C3>C3_th, delete previous checkpoint, delete logs and UIDs, interval=next

So, per interval each mh saves checkpoint, logs and UIDs of single checkpoint interval.

Theorem 3: Checkpoint traverses only through trusted mobile hosts

Proof: Following two lemmas help to prove this theorem.

Lemma 1: Only Trusted mobile hosts can be selected as cluster heads and gateway nodes.

Proof: If there is n number of mobile hosts in a cluster then the mobile host having highest trust value will be selected as cluster head and mobile hosts having next higher trust value/values will be selected as gateway node/nodes. If trust value of current cluster head deceases due to reduced battery power or unavailability in the network etc., then it will be replaced by another mobile host having highest trust value currently.

Lemma 2: Saved last checkpoint copy of failed mobile host is forwarded through only cluster heads and gateway nodes to the failed host for its recovery

Proof: Last checkpoint copy of any mobile host is saved in cluster head. When it fails after some inter-cluster movements, the cluster head in which it will recover communicates with other cluster heads to find and retrieve its last checkpoint. Then the checkpoint saving cluster head forwards it to the communicating cluster head through intermediate cluster heads and gateway nodes.

6. Performance Analysis

We simulate the whole MANET environment like clusters, cluster heads, gateways, cluster members, movement, cluster change, communication through message passing, logging, checkpointing, failure, recovery all these non-deterministic events randomly using C programming. The mobility model considered here is random way point model [18]. We use rand () function in C to generate join, leave, send and receive function randomly. We have implemented random movement, handoff, trust model computation message sending and receiving, logging, checkpoint, failure and rollback recovery of mobile hosts to get different parameters for the performance analysis of the proposed algorithm. We get all this computation time using clock () function. By using clock() function we get the time to take checkpoint and log, time required to decide whether a node is trusted and distrusted, time required to cluster change and failure time by using clock(). We consider following parameter set to show performance analysis of proposed mobility based checkpointing and trust based rollback recovery technique. The parameter values which we use in our simulation are represented in tabular form in below:

Table 3. Performance Analysis parameters value

Parameter	**value**
Checkpoint Size	2-256 KB
Log Size	50 B
computation message size	50 B
coordination message size	2.5 B
UID size	2.5 B
time to transfer checkpoint per hop through wireless channel	0.08s

time to transfer log or computation message per hop through wireless channel	0.002s
time to transfer coordination message or UID of a log.	0 .0001s
Energy capacity of a node	1800J
Availabity	10-100%
Mobility rate	0.40-0.48
Channel bandwidth	1 MB
Failure rate	0.02-0.08
Cluster change threshold	3-20
Recovery probability	0.4-0.99

c3 vs. failed node's recovery time (assuming mobile host fails when c3 = c3_th) :

A cluster member node moves from one cluster to another cluster, c3 increases by 1, distance between old and new clusterhead increases by 1 hop. Here we assume that a node fails when c3 = (c3_th) i.e. just before c3 exceeding (c3_th), the condition to save checkpoint of current state. At this point recovery information, logs have to be retrieved from (c3_th) number of cluster heads and checkpoint will be transferred from the cluster head which is at a distance of (c3_th) number of hops from recovery clusterhead. Thus log transfer cost and checkpoint transfer cost both will be maximum. So it is justified that if checkpoint is saved based on c3_th then maximum recovery cost can be calculated beforehand. This will help to set the value of (c3_th) as per application.

Recovery time = cost to transmit recovery request message to (c3_th) no. of cluster heads + cost to transfer logs saved in c3_th no. of cluster heads + cost to transfer last checkpoint from the cluster head which is at a distance of c3_th hops

$= (Cm_{co}* (c3\_th) + Clog\_transfer * (c3\_th) + Ccheckpoint\_transfer*(c3\_th) \text{ hops}) \text{ unit}$

$= ((Cm_{co}+ Clog\_transfer + Ccheckpoint\_transfer) * (c3\_th))\text{unit}$

$= ((0.0001 + 0.002 + 0.08) * (c3\_th)) \text{ unit}$

$= (0.0821 * (c3\_th)) \text{ unit}$ -------------------- (3)

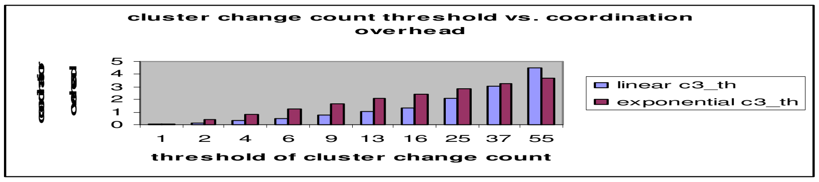


Figure 2. maximum coordination overhead at the time of recovery varies with cluster change count threshold

A mobile node fails when c3= (c3_th), just before saving checkpoint. Hence number of cluster heads where the node's recovery information are scattered is c3_th. For simplication we consider that distance between last cluster head where the ID of the cluster member node that saves backup copy of last checkpoint of failed node is saved and the cluster head where the failed node recovers is (c3_th) hops. Here checkpoint transfer cost is bounded by (c3_th).

Number of cluster member nodes vs. coordination overhead :

Coordination message for Checkpoint-recovery

= recovery message$_{mh\text{-}CCH}$ + transfer_checkpoint_message$_{CCH\text{-}CH}$ + transfer_checkpoint_message$_{CH\text{-}mh}$
= m_{co}+ CH* m_{co} + m_{co} = $m_{co}(1+CH+1) \approx$ CH (for high value) --------------------------(4)

coordination message for Log recovery
=log recovery message$_{mh\text{-}CH}$+ transfer log message$_{CH\text{-}CH}$
$\approx (1+CH)*m_{co} \approx$ CH, hence total coordination message = CH + CH = 2*CH

Total coordination message ∞ CH ------------------------ (5)

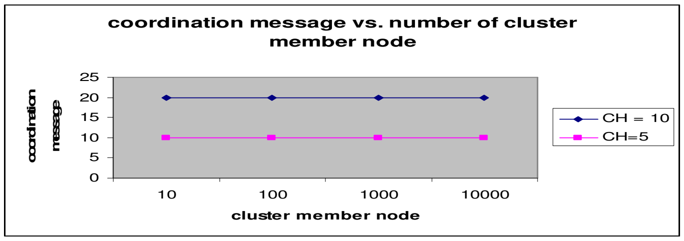


Figure 3. recovery coordination message varies with cluster head irrespective of number of cluster member nodes of a cluster

In figure 3 it is shown that coordination among different nodes at the time of recovery of a failed node is restricted to cluster head level. Coordination message overhead does not depend on number of cluster member nodes.

Next, we compare recovery probability by considering selection of CH and GW based on trust model and considering selection of CH and GW based on individual factor like failure rate, available battery power, percentage of availability and recommendation. We have calculated and plotted recovery probability vs. varying individual factor among the four mentioned above while the other three is constant,

Case1: Selection of CH and GW based on failure rate.
Case2: Selection of CH and GW based on percentage of availability of CM.
Case3: Selection of CH and GW based on available battery power of CM.
Case4: Selection of CH and GW based on recommendation from neighbour CM.
Case5: Selection of CH and GW based on Trust Model.

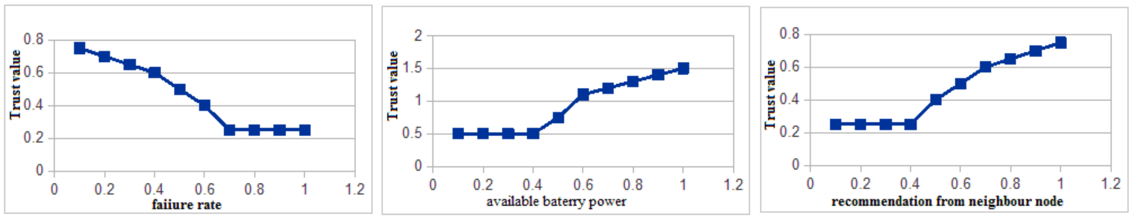
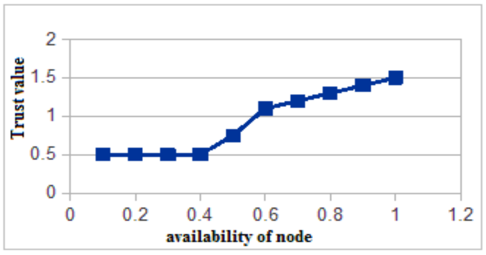


Figure 4.(a) failure rate of node vs. Trust value; (b)available battery power of node vs. Trust value;(c) recommendation from neighbour node vs. Trust value

In figure.4.(a) we see that if failure rate of node increases trust value will decrease.So, failure rate is inversely proportional to trust value. In figure 4.(b) we see that if availability of baterry power increases trust value will increase.So, it is directly proportional to trust value. In figure

4.(c) we see that if recommendation from neighbour node increases trust value will increase.So, it is directly proportional to trust value.



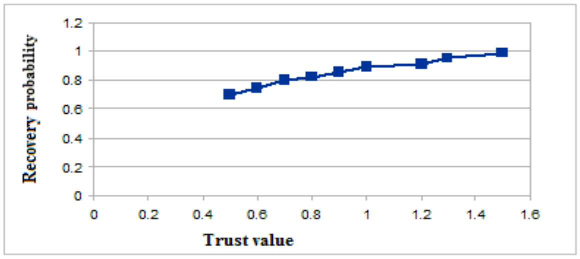
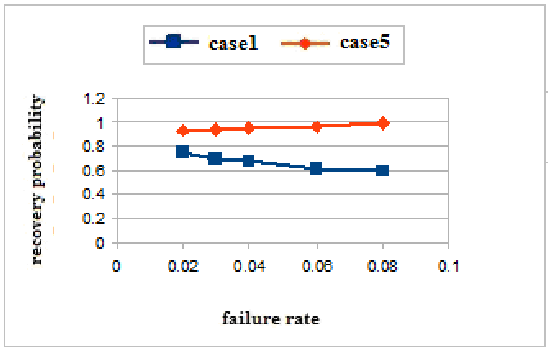


Figure 5.(a) Availablity of node vs. Trust value; (b) Trust value vs. Recovery probability

In figure 5.(a) we see that if availability of node increases trust value will increase. So,availability of node is directly proportional to trust value.In figure 5.(b)we see that recovery probability increases with trust value. we get 0.99 as highest recovery probability. In all the comparison we consider failure rate 0.08. At this failure rate we consider available battery power, percentage of availability and percentage of recommendation will be 100%.



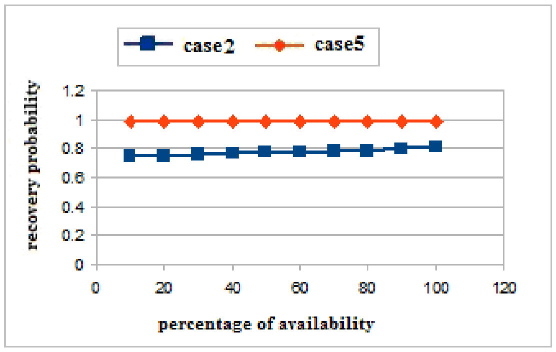
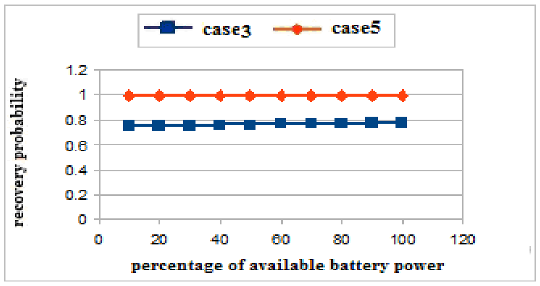


Figure 6.(a) Failure rate vs. recovery probability considering case1 and case5 ; (b) Percentage of availability vs. recovery probability considering case2 and case5

In figure 6(a) we see that if failure rate of CH and GW is increased recovery probability will decrease because if failure rate of CH and GW is high then checkpoint and log may not be transferred all time to the destination cluster. Therefore the recovery probability decreases as the failure rate increases. In the above figure we compare between case5 and case1.We see that recovery probability of case5 is more than case1. In figure 6(b) we see that if percentage of availability of CH and GW is increased recovery probability will increase because if percentage of availability of CH and GW is high then failure rate will be low. In above we have already discussed failure rate is inversely proportional to recovery probability. As percentage of availability is inversely proportional to failure rate, percentage of availability is directly proportional to recovery probability. In the above figure we compare between case5 and case2.We see that recovery probability of case5 is more than case2.



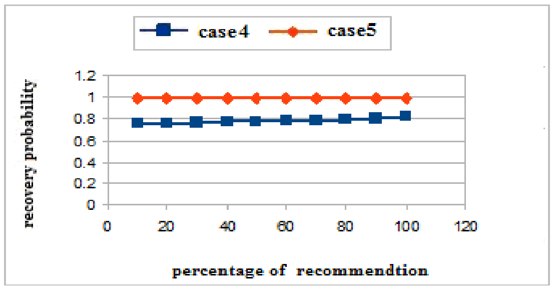


Figure 7.(a) Percentage of available battery power vs. recovery probability considering case3 and case5; (b) Percentage of recommendation vs. recovery probability considering case4 and case5

In figure 7(a) we see that if available battery power of CH and GW is increased, recovery probability will increase because if available battery power of CH and GW is high then failure rate will be low. We have already discussed failure rate is inversely proportional to recovery probability. As available battery power is inversely proportional to failure rate, available battery power is directly proportional to recovery probability. In figure 7.a, we compare between case5 and case3.We see that recovery probability in case5 is more than case3.

In figure 7(b) we see that if percentage of recommendation of CH and GW is increased, recovery probability will increase because if percentage of recommendation of CH and GW is high then failure rate will be low. We have already discussed that failure rate is inversely proportional to recovery probability. As percentage of recommendation is inversely proportional to failure rate, percentage of recommendation is directly proportional to recovery probability. In figure 7.b, we compare between case5 and case4.We see that recovery probability of case5 is more than case4.

After analyzing the above comparisons we get the following results.

Table 4. Performance analysis result

desired recovery probability	**Highest Recovery Probability we get from our experimental result**				
	Case1	Case2	Case3	Case4	Case5
1	.75	.81	.78	.82	.99

After analyzing table 4, we see that we get highest recovery probability in case5. So, selection of CH and GW based on proposed trust model gives high recovery probability which is desired in any fault tolerant system. We get all the above values through our simulation result. In case1, CH and GW get selected based on only failure rate without considering available battery power, availability and recommendation. So, after failure when checkpoint is transferred through GW or CH, there is a possibility of failure of CH and GW due to low available battery power or availability or poor recommendation. So, checkpoint and log may not be successfully transferred all the time to the destination. Hence recovery probability may be very low. This may be true in case2, case3 and case4 mentioned above. But in case5, we select CH and GW considering above four factors so that chances of failure of CH and GW will be less. So we can get maximum 99% recovery probability using proposed trust model.

7. Conclusion

In this work we propose a mobility based checkpointing and trust based rollback recovery algorithm combined with message logging to provide fault tolerance in cluster based mobile ad hoc network. Mobility based checkpointing limits recovery time and trust based recovery increases recovery probability of failed mobile hosts. Each cluster member keeps a count of its change of clusters and if this count exceeds a predefined threshold, then the mobile host will save checkpoint. What makes more challenging the task of checkpointing in MANET is lack of stable storage. In MANET, checkpoint and log placement is also a very trivial issue. Checkpoint traversal only through trusted mobile hosts ensures successful recovery of failed mobile host. Failure prone components are security attack prone also. Moreover insecurity leads to distrust and vice versa. So, security of checkpoints in mobile hosts can be added as a factor to calculate trust value of a mobile host. Proposed trust model will be extended towards that.

References

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

[2] C. Men, Z. Xu , D. Wang , (2007)"An Efficient Handoff strategy for Mobile Computing Checkpoint System", EUC 2007, LNCS 4808, pp. 410–421.

[3] F. Quaglia, B. Ciciani , R. Baldoni (2006)"Checkpointing Protocols in Distributed Systems with Mobile Hosts: a Performance analysis",Workshop on Fault-Tolerant Parallel and Distributed Systems, pages 742-755.

[4] R. Prakash, M. Singhal, (1996)"Low Cost Checkpointing and Failure Recovery in Mobile Computing Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 7, pp:1-38.

[5] G.Hong, S.J.Ahn, S.C.Han, T. Park, H.Y. Yeom, Y.Cho,(2000) " Kckpt: Checkpoint and Recovery Facility on UnixWare Kernel" [J], Computers and Applications, pp. 303-308.

[6] P. Basu, N. Khan, and T.D.C. Little, (2001)"A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks," in Proc. IEEE ICDCSW' 01, Apr., pp. 413–418.

[7] J.H.Cho, A.Swami (2011)"A Survey on Trust Management for Mobile Ad HocNetworks", Communications Surveys & Tutorials, IEEE, Computer & Information Science,Volume: 13, pp. 562 – 583.

[8] S.Biswas,S.Neogy,(2012)"Checkpointing and Recovery using Node Mobility Among Clusters in Mobile Ad hoc Network", , Springer, NECOM pp. 447-456.

[9] P.K. Jaggi, A.K. Singh,(2011) "Staggered Checkpointing and Recovery in Cluster Based Mobile Ad Hoc Networks ", Advances in Parallel Distributed Computing Communications in Computer and Information Science, , Volume 203, pp. 122-134

[10] T. Ying., T. Juang, M.C. Liu, (2002) "An efficient asynchronous recovery algorithm in wireless mobile adhoc networks", Journal of Internet Technology Special Issue on Wireless Internet: Applications and Systems¨. Vol 3, No.2, pp 147-155

[11] Y. Sangho, J. Heo, Y. Cho, J. Hong,(2006) "Adaptive Mobile Checkpointing Facility for Wireless Sensor Networks", LNCS 3981, pp. 701-709.

[12] X.Zhao, Z.You, Z. Zhao, D.Chen, F.Peng,(2010)"Availability Based Trust Model of Clusters for MANET" Service Systems and Service Management (ICSSSM), 7th International Conference on,pp: 1 – 6.

[13] A. fathi, H. taheri,(2010)" Enhance Topology Control Protocol (ECEC) to Conserve Energy based clustering in Wireless Ad Hoc Networks",ICCSIT, 3rd IEEE International Conference , Volume: 9,pp. 356 – 360.

[14] J. Luo, X.Liu , M.Fan,(2009)"A trust model based on fuzzy recommendation for mobile ad-hoc networks", Journal Computer Networks: The International Journal of Computer and Teleco-mmunications Networking , September,Volume 53,pp. 2396–2407.

[15] R. Manoharan, S. Mohanalakshmie,(2011)"A Trust Based Gateway Selection Scheme for Integration of MANET with Internet" , IEEE-International Conference on Recent Trends in Information Technology,MIT, Anna University, Chennai, ,pp543-548.

[16] E.N. (Mootaz) Elnozahy, L.Alvisi, Y.Wang and D.B. Johnson, (September 2002), "A Survey of Rollback Recovery Protocol in Message Passing System"ACM Comput. Surv., Vol. 34, No. 3. pp. 375-408.

[17] A.Jøsang,(1999)"An Algebra for Assessing Trust in Certification Chains", In J.Kochmar, editor, Proceedings of the Network and Distributed Systems Security (NDSS'99) Symposium, The Internet Society, pp.1-10

[18] F.Li, J.Wu, (2007) "Mobility Reduces Uncertainty in MANETs", INFOCOM.26th IEEE International Conference on Computer Communications. IEEE, pp.1946 – 1954.

[19] T. Camp, J. Boleng, and V. Davies, (2002,)"A survey of mobility models for adhoc network research", Wireless Comm. & Mobile Comp. (WCMC), vol. 2, no. 5,pp: 483-502.

Authors

Suparna Biswas obtained M.E. from Jadavpur University. She is currently working as an Assistant Professor in the Dept. of Computer Science & Engg., West Bengal University of Technology. Her areas of research interests are Fault Tolerant Mobile Computing, Software Engineering etc.

Sarmistha Neogy is an Associate Professor in the Department of Computer Science & Emgineering, Jadavpur University, at present and is in teaching profession since last eighteen years. She has been an active researcher in the areas of distributed systems, fault tolerance, mobile computing and security in wireless networks.

Priyanka Dey completed her B.Tech in Information Technology from Techno India College of Technology. Presently she is pursuing her M.Tech in Software Engineering from Dept. of Computer Science & Engg., West Bengal University of Technology. Her research interest is "Fault tolerance in mobile computing".