5. Trusted Checkpoint-Recovery in MANET

5.1 Introduction

Mobile Ad hoc Network (MANET) is based on cooperation of autonomous mobile hosts which acts as routers also. For successful routing, all intermediate mobile hosts should receive and forward packets properly. If any of the intermediate mobile host’s behavior is selfish or malicious, then either the packet may simply be dropped and not forwarded or modified. Since these are not desired, packet may be encrypted by the sender mobile host and sent. Checkpointing and recovery technique itself adds some computational overheads. Cryptography of checkpoints introduces some problems: a) encryption and decryption overheads b) secret key sharing over wireless network which is vulnerable to security attacks. Additional overheads due to cryptography further increases overhead of checkpointing - recovery technique and it will become impractical to implement it in resource constrained, infrastructure less, dynamic mobile ad hoc network. A mobile host can be judged to be a trusted entity with a belief that it will behave properly. A trusted mobile host will forward data packet without fail. The other desirable properties of a trusted mobile host are that it should have sufficient resources and low failure rate. The question that arises now is what factors should be considered while measuring trust value of a mobile host.

In this chapter two different trust models have been proposed for trust estimation and trust status assessment of a mobile host. First trust model considers the four factors to calculate trust value of a mobile host: availability, failure rate, available battery power, recommendation from neighbor mobile hosts. Availability of a mobile host in the network is considered to ensure that intermediate mobile hosts do not disconnect before forwarding checkpoint to next MH. Failure rate is considered so that the mobile host does not fail while forwarding checkpoint. Available battery power is important to ensure that intermediate mobile host does not get switched off while
forwarding. Though these factors represent current status of a MH, recommendation from other mobile hosts is taken to find its past performance as a participating mobile host in checkpoint-recovery process. The second trust model proposed is based on the concept of Ant Colony Optimization Technique. The checkpoint is considered as ant and if it is forwarded successfully by a mobile host, then pheromone concentration on this mobile host increases otherwise decreases. For traversal of another checkpoint, the set of intermediate mobile hosts between checkpoint node and recovery node will be selected based on pheromone concentration. Using this concept, three factors considered to calculate trust value and evaluate trust status of mobile hosts are: successful response, failure rate and available battery power. A mobile host responds successfully if it is available in the network. The response is taken as “goodwill” of the mobile host. Failure rate and available battery power are considered for the reasons explained above in the other trust model. Important properties of trust for MANET nodes are considered to be [5]:

a) **Dynamic**: Trust of a mobile host is not a constant quantity and should be evaluated continuously. Evaluation of trust is limited spatially and locally.

b) **Subjective**: Due to dynamically changing environment, a MH may be evaluated to belong to different trust levels at different times. Hence, trust in MANET is subjective. For example, availability of a mobile host is a criteria to evaluate trust. When a mobile host is connected to network, its availability is 1, otherwise 0. Accordingly, its trust value will also change.

c) **Not Transitive**: Let us consider, A will send a data packet to D via B, C. Now A will want to ensure that B and C must be trusted. A knows B to be trusted, B knows C to be trusted. Does this imply that A will trust C? A may take recommendation from B about C and then calculate trust value of C, which is an indirect observation of A about C.

d) **Asymmetric**: In heterogeneous MANET, some MHs may be well equipped with higher level of resources and others may have lower level of resources. The former ones will however not trust later ones.

e) **Context dependent**: A may trust B as a data forwarding node but not for some other functionality. So, depending on the context, one MH may or may not trust other MH/s.
MANET is an ever changing, unstable, resource constrained, dynamic environment in which the knowledge or information gathered of a mobile host may not always be perfect. So, the assessment of trust of a mobile host may be considered as an opinion which will indicate the degrees of ‘belief’, ‘disbelief’ or ‘uncertainty’ of MHs. This can be expressed mathematically [68] as follows:

\[ b + d + u = 1, \quad \{b, d, u\} \in [0,1]^3 \]

Application of MANET includes military communication, emergent operations, personal electronic device networking, and civilian applications like an ad-hoc meeting or an ad-hoc classroom. All these applications specifically military communications need fault tolerance and security of data. Cryptography in MANET is infeasible due to increased overheads. Hence trust is used to ensure security.

5.1.1 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 and topology changes dynamically. Here recovery for both transient and crash failure of MHs is considered. It is assumed that the mobile host that saves checkpoint and the cluster head that carries the copy will not fail simultaneously.

5.2 Mobility Based Checkpointing and Trust Based Recovery in MANET

Proposed work is a mobility based checkpointing and trust based rollback recovery algorithm for MANET. The network considered here is cluster based hierarchical MANET [69]. 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. The
checkpoints of cluster member mobile hosts of a cluster are generally saved in cluster head. Now the mobile host (MH) moves across different clusters and may fail in other cluster. 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. If any of these MHs fail, then the checkpoint may be lost and the recovery process will be terminated. So, the challenge is to select a set of MHs through which checkpoint can be successfully transferred without any fail. This can be avoided if it is ensured that the checkpoints and logs are forwarded only through trusted MHs. Trust value of a mobile host is calculated based on following four factors: failure rate, availability, available battery power and recommendation from neighbor MHs. A MH having low failure rate and high available battery power, will be highly available in network ensuring successful execution of any assigned task enhancing positive recommendations from neighbor MHs. A threshold of trust value has been defined. A mobile host is considered as trusted if its trust value is greater than a predefined threshold. Simulation results show that proposed algorithm achieves low recovery cost and high recovery probability of failed mobile hosts.

5.2.1 Problem Identification

After analyzing related works described in [4][43][44][70][71] following problems are identified. To provide solutions to these problems, a new trust model is proposed. Identified problems are:

Cluster Head has limited memory space: In [43], 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 the authors do not specify number of cluster members in a cluster so that cluster head can save all cluster members’ checkpoints.

Random mobility of cluster head and cluster member nodes: Above mentioned related works do not consider mobility of cluster members and cluster heads.
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 checkpointing and recovery process do not fail.

Trust value calculation based on a single factor: In [4][70][71] cluster head is selected based on only a single factor. In [70], cluster head is selected based on a mobile host’s availability. In [71], cluster head is selected on the basis of only available battery power. In [4], 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 proposed work following issues are taken into consideration.

Limited cluster members in a single cluster: If it is considered that a specific memory space of a cluster head is allocated to save checkpoints of its cluster member nodes then maximum number of cluster members of a cluster is defined by dividing allocated memory space of a cluster head to save copy of checkpoints of its cluster members by size of a checkpoint.

Inter-cluster movement based checkpointing: Mobile hosts save checkpoints based on their inter-cluster movement.

Only trusted MHs 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.

5.2.2 Proposed work

Here proposed trust model is described. Checkpointing is mobility based and recovery is trust based.

5.2.2.1 Checkpointing and Recovery Technique

During a checkpoint interval, mobile host sends and receives computation messages to and from other mobile hosts. A mobile host saves log of sent computation message and forwards it to current cluster head. A cluster head saves logs of its cluster members, coordinates recovery of its failed cluster member hosts besides own computations. A mobile host increments ‘cluster-change-count’ by 1 each time it leaves a cluster and joins
another. Each MH 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 saves last checkpoint of failed MH 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 and recover consistently.

![Diagram](image_url)

**Figure 5-1 Working Example of the Network Considered in our Work**

Above figure represents the cluster based mobile ad hoc network considered in our work. Following abbreviations are used to illustrate the working example:

- C = cluster, CM = cluster member, CH = cluster head, GW = gateway, c3 = cluster change count, c3_th = cluster change threshold

In Figure 5-1, 4 clusters C1, C2, C3, C4 are shown. 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 C1, C2, C3 and C4 have cluster heads (CH): CH1, CH2, CH3 and CH4 respectively. Each cluster head of one cluster is connected to cluster head of another cluster through gateway nodes (GW): GW1, GW2 and GW3. Each cluster member will take checkpoint when threshold of cluster-change-count (c3_th) = 2. The kth member of ith cluster is denoted by CM_{k,i}. For example, CM_{1,1} initially takes checkpoint at C1 and saves checkpoint and logs in CH1. So after visiting C2, C3 when CM_{1,1} will reach C4, its cluster change count (c3) will be 3 which exceeds c3_th. So CM_{1,1} will save checkpoint and logs in CH4. After taking checkpoint in C3, two cases may happen:
Case 1: CM₁,₁ may fail before saving checkpoint CH₄

CM₁,₁ fails before saving checkpoint in CH₄, and later reconnects to CH₄ for recovery. CH₄ broadcasts this recovery message to all CHs to find the CH in which CM₁,₁’s last checkpoint and logs are saved. CH₁ replies to CH₄. CH₁ will send checkpoint and logs through CH₁ - GW₁ - CH₂ - GW₂ - CH₃ - GW₃ - CH₄. CH₄ then sends checkpoint and logs to CM₁,₁. CM₁,₁ will rollback to this checkpoint and replay the logs. This mobility based checkpointing and log based rollback recovery is consistent.

Case 2: CM₁,₁ may save checkpoint and logs in CH₄

CM₁,₁ saves checkpoint and logs in CH₄ and sends message to CH₄ to delete last saved checkpoint and logs which is saved in CH₁. CH₄ sends message to CH₁ to delete checkpoint and logs of CM₁,₁. After receiving message CH₁ deletes checkpoint and logs of CM₁,₁ and acknowledges CH₄.

This recovery technique does not consider failure probability of intermediate MHs through which the checkpoint is transferred to the failed mobile host (Case 1). So if any checkpoint forwarding MHs fails, checkpoint transfer will be delayed or unsuccessful depending on the type of failure. Any of these is 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 cluster head will save a copy of checkpoint. Then the cluster head will delete the checkpoint. Within stipulated time, if successful recovery message does not come or failed recovery message comes, the cluster head will again send a copy of checkpoint and wait for reply. Reasons of this could be failure of forwarding MHs. In worst case, this may continue repeatedly causing resource consumptions of cluster head and intermediate forwarding mobile hosts.

ii) Checkpoint is forwarded only through ‘trusted MHs’: If it can be ensured that checkpoint forwarding is done through only ‘trusted MHs’ that would not fail during recovery process of a MH then recovery-process would not fail. To ensure that, current MH must be sure about trust status of next MH. So, trust value of next MH has to be calculated and that can be done using the proposed trust model.

The above second alternative solution is much more acceptable because checkpoint forwarding through trusted MHs will ensure that checkpoint will reach the failed mobile
host. 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 MHs.

### 5.2.2.2 Proposed Trust Model

Trust of cluster member is a belief that the cluster member will resist different types of attack. A cluster member can be either trusted or distrusted. Due to lack of communication or improper knowledge sometimes it may be impossible to decide whether a cluster member is ‘trusted’ or ‘distrusted’. In this case cluster member is ‘uncertain’. In this work ‘belief’ is ‘trusted’, ‘disbelief’ is ‘distrusted’. There are various methods of trust value evaluation [4][71], trust based gateway selection [72] and removal of uncertainty [46] in a system. In proposed algorithm trust value is calculated based on values of availability, recommendation from neighbor nodes, failure rate, and unused energy per mobile host.

**Availability:** This is a statistical data accumulated during a MH’s communications with other MHs. Each MH estimates its neighbor’s availability based on its accumulated observations using Bayesian inference as discussed in [46]. Bayesian inference is a statistical inference in which evidence or observations are used to update statistics. Beta distribution, \([\text{Beta}(a, b)]\) needs only two parameters that are continuously updated as communications are made. When a new communication is made, ‘a’ is updated for successful communication, otherwise ‘b’ is updated. Hence beta distribution may be used here. Every MH always maintains experience statistics record of its neighbor mobile hosts. Every mobile host always maintains availability record of its neighbor MH. When availability of a MH is calculated, availability record from its neighbor hosts is taken.

**Algorithm to evaluate availability (A)**

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

**Failure rate \((F_r)\):** It is an important factor to decide whether a mobile host 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.
**Available Energy (E):** If a MH has high energy then its probability of disconnection from network will be low. So, MH will be more reliable.

**Recommendation from neighborMHs (r):** Many a times, recommendation helps in arriving at decisions. Many trust models are already implemented based on this. So, this parameter is included here also. A mobile host will give recommendation about other host based on its availability, failure rate and available energy.

For example, suppose A will have to give recommendation of B. In that case, B has to first share its failure rate, availability and available energy with A. Then A checks availability by using algorithm as already discussed above. Then recommendation can be set as given below in Table 5.1.

**Table 5.1 Set Recommendation Value on Different Factors**

<table>
<thead>
<tr>
<th>Recommendation(r)</th>
<th>Failure rate (Fr)</th>
<th>Availability(A)</th>
<th>Available Energy (E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt; 0.5</td>
<td>&gt; 0.5</td>
<td>&gt; 0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>= 0.5</td>
<td>= 0.5</td>
<td>= 0.5</td>
</tr>
<tr>
<td>0</td>
<td>&gt; 0.5</td>
<td>&lt; 0.5</td>
<td>&lt; 0.5</td>
</tr>
</tbody>
</table>

We consider 0.5 to be the threshold of all three parameters. In equation 3, $T_{val\_th} =$ threshold of trust value, $Fr\_th =$ threshold of failure rate, $A\_th =$ threshold of availability, $E\_th =$ threshold of energy.

So, Trust_value_threshold ($T_{val\_th}$) = $(Fr\_th \times r) + (A\_th \times r) + (E\_th \times r)$  
= $(0.5 \times 0.5) + (0.5 \times 0.5) + (0.5 \times 0.5)$  
= $0.25 + 0.25 + 0.25 = 0.75$

By using following algorithm, evaluation of trustworthiness of a mobile host is done:

if ( (trust_value >= trust_value_threshold) && A >= 0.5 && Fr <= 0.5 && E >= 0.5 )
mobile host is ‘trusted’;
else { if ( A = = 0 ) mobile host is ‘uncertain’;
    else mobile host is distrusted ; }

If a MH is evaluated to be uncertain, failure rate of the node is checked. If failure rate is less than 0.5, mobile host will be considered as trusted, otherwise distrusted. For example, suppose the trust_table of three mobile host of a cluster is as follows:

123
Table 5.2 An Example of Leveling of MH after Computing Trust Value

<table>
<thead>
<tr>
<th>MH</th>
<th>E</th>
<th>A</th>
<th>R</th>
<th>Tr_val</th>
<th>Level of MH</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.2</td>
<td>0.2</td>
<td>0.7</td>
<td>(1,0,1)</td>
<td>0.9 &gt; 0.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Trusted</td>
</tr>
<tr>
<td>B</td>
<td>0.2</td>
<td>0</td>
<td>0.7</td>
<td>(1,0,1)</td>
<td>0.9 but A==0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Uncertain</td>
</tr>
<tr>
<td>C</td>
<td>0.6</td>
<td>0.02</td>
<td>0.6</td>
<td>(0,0,1)</td>
<td>0.6 &lt; 0.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>distrusted</td>
</tr>
</tbody>
</table>

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

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 gateway node. A mobile host once trusted may not be trusted forever because its battery power may reduce, its availability in the network may be poor, its failure rate may increase with time and if any or all of these are true, then other MHs may not recommend it. Then a trusted MH may become distrusted or uncertain. If due to reduced trust value, or random mobility or any other reason, cluster head or gateway nodes get changed, these MHs will transfer all the saved checkpoints, logs, log records to the newly selected cluster head or gateway nodes respectively. 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.

5.2.3 Algorithm

1. Each mobile host sends a beacon message to own cluster head at regular interval to convey that it is connected. This ensures authentication of MHs.
2. Sender MH saves log of computation message, forwards it to current cluster head. Cluster head saves log and forwards it towards destination.
3. Receiver MH receives computation message, updates dependence list, and sends log records to gateway node.

4. If a MH moves then in cluster based ad hoc network, each mobile host increments cluster_change_counter (c3) each time a mobile host changes cluster.

5. Before the mobile host leaves a cluster head, sends different data structures for example dependence list, list of CHs traversed etc. saved along with a leave message to current cluster head.

6. The mobile host next joins another cluster head.

7. if \( c3 > c3_{th} \), go to step 8
   else repeat steps 1 to 6

   // Checkpoint Procedure //

8. Mobile host invokes checkpoint procedure
   8.1 The MH takes a local checkpoint
   8.2 Searches for an immediate neighbor MH having highest available memory and saves a back up copy of checkpoint in it. Sends ID of neighbor MH in which backup copy of its checkpoint is saved to its current cluster head.

9. The MH sends a message “delete_log_of_previous_interval” to its current cluster head that forwards this message to the cluster heads that are saved in the list of CHs traversed so far.

   9.1 If any gateway node finds a match between logs to be deleted and the Log records saved in it, the gateway node sends a message, “do_not_delete_the_log” to the cluster head.
   9.2 The cluster head forwards log to the gateway node and deletes own copy.
   9.3 Gateway node saves log.

   // Recovery Procedure //

10. The MH fails, its last checkpoint to be recovered from the MH itself or cluster head

   10.1 Last checkpoint can be recovered from failed host transient failure occurs:
      The MH rolls-back upto last checkpoint.

   10.1.1 The failed host has sent computation messages only before failure and after last checkpoint:
‘Replay_log’ message is sent to current cluster head that checks if the logs are saved in its memory, if yes, replays log to corresponding receiver MHs, else broadcasts ‘replay_log’ message to the cluster heads saved in failed MH’s ‘list of traversed CHs’.

10.1.2 The failed host has received computation messages only before failure and after last checkpoint:
Receive_computation message is sent to current cluster head that checks if the log records are saved in its memory, the receiver MH receives computation messages accordingly.

10.1.3 The failed host has both sent and received computation messages before failure and after last checkpoint:
Do tasks mentioned in 10.1.1 and 10.1.2.

10.2 Last checkpoint needs to be recovered from cluster head:
Current CH sends recovery_message of the failed MH to the CH that saved the failed MH’s last checkpoint. CH forwards copy of checkpoint of failed host saved with it following the same path through which the recovery_message has been sent to it.

11. The failed host rolls back upto last checkpoint and recovers.

5.2.3.1 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 logs of each sent computation message \( m_c \). TRUE for \( \forall MH \) if (MH fails without saving checkpoint)

sent \( m_c \) is retrieved from saved logs, TRUE for \( \forall \) sent \( m_c \)

Lemma 2: There is no lost message.
Proof: Save log record of each received \( m_c \). TRUE for \( \forall MH \) if (MH fails without saving checkpoint)

Determinant of received \( m_c \) is retrieved as per saved information, TRUE for \( \forall MH \)
Lemma 1 satisfies hard constraint and lemma 2 signifies soft constraint of consistent recovery. Thus theorem 1 is proved.

**Theorem 2: 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 are 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/s will be selected as gateway node/s. If trust value of current cluster head decreases 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. Since trust values are evaluated continuously, the detection is possible.

**Lemma 2: Saved last checkpoint copy of failed mobile host is forwarded through a trusted path 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 the MH’s last checkpoint. Then the checkpoint saving cluster head forwards it to the communicating cluster head through intermediate cluster heads and gateway nodes since the routing here will only consist of cluster heads and gateway nodes as per hierarchical routing protocol techniques.

### 5.2.4 Performance Analysis

MANET environment along with all different types of nodes (Cluster head, cluster member, gateway) and their movement, communication through message passing, logging, checkpointing, failure, all these non-deterministic events are simulated using C programming. The mobility model considered here is random way point model [73]. Join, leave, send and receive functions of MHs are randomly generated. Different computation times are obtained by implementation of the algorithm. Using clock () function time to take checkpoint and log, time required to decide whether a MH is trusted and distrusted, time required to cluster change and failure time etc. are obtained. Following parameter
set is used for performance analysis of proposed mobility based checkpointing and trust based rollback recovery technique.

### Table 5.3 Performance Analysis Parameters Value

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkpoint Size</td>
<td>2-256 KB</td>
</tr>
<tr>
<td>Log Size</td>
<td>50 B</td>
</tr>
<tr>
<td>Computation message size</td>
<td>50 B</td>
</tr>
<tr>
<td>Coordination message size</td>
<td>2.5 B</td>
</tr>
<tr>
<td>Log record size</td>
<td>2.5 B</td>
</tr>
<tr>
<td>Time to transfer checkpoint per hop through wireless channel</td>
<td>0.08s</td>
</tr>
<tr>
<td>Time to transfer log or computation message per hop through wireless channel</td>
<td>0.002s</td>
</tr>
<tr>
<td>Time to transfer coordination message</td>
<td>0.0001s</td>
</tr>
<tr>
<td>Energy capacity of a node</td>
<td>1800J [74]</td>
</tr>
<tr>
<td>Availability</td>
<td>10-100%</td>
</tr>
<tr>
<td>Mobility rate</td>
<td>0.40-0.48</td>
</tr>
<tr>
<td>Channel bandwidth</td>
<td>1 MB</td>
</tr>
<tr>
<td>Failure rate</td>
<td>0.02-0.08</td>
</tr>
<tr>
<td>Cluster change threshold</td>
<td>3-20</td>
</tr>
<tr>
<td>Recovery probability</td>
<td>0.4-0.99</td>
</tr>
</tbody>
</table>

- **Cluster_change_count 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 cluster head increases by 1 hop. Here it is assumed 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 cluster head. 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 estimated earlier. This will help to set the value of (c3_th) as per application requirement.

- **Recovery time**

A mobile host 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 simplification it is considered 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).

Recovery time = cost to transmit recovery request message to (c3_th) number of cluster heads + cost to transfer logs saved in c3_th number of cluster heads + cost to transfer last checkpoint from the cluster head which is at a distance of c3_th hops

= (cost to transfer coordination message + cost to transfer logs + cost to transfer checkpoint) * c3_th) unit

= ((0.0001 + 0.002 + 0.08) * (c3_th)) unit

= (0.0821 * (c3_th)) unit  \[ \text{------------------------ (3)} \]

So, coordination overhead that is measured in terms of recovery cost, varies with threshold of cluster change count. This is observed in Figure 5-2 in which coordination overhead is calculated by varying c3_th both linearly and exponentially within a range from 1 to 55. Coordination overhead varies according to the variation of c3_th.

![Figure 5-2 Maximum Coordination Overhead at the Time of Recovery Varies with Cluster_Change_Count Threshold](image)

- **Coordination message required to retrieve last checkpoint of failed MH**

129
Number of coordination messages required to retrieve last checkpoint of failed MH

= to send recovery message from failed MH to the cluster head in which the MH will recover + to broadcast coordination message from recovery cluster head to all cluster heads + to send reply from last checkpoint saving cluster head to failed MH.

= (1 + (CH - 1) + 1) ≈ (1 + CH) ≈ CH (ignoring 1)

-------------------------(4)

- **Coordination messages for Log recovery**

= log recovery message from MH to current cluster head + transfer log message from current CH to all CHs

≈ (1 + CH) * mco ≈ CH, hence total coordination message = CH + CH = 2*CH

Total coordination message ∝ CH

-------------------------(5)

![Figure 5-3 Recovery Coordination Message vs. Number of Cluster Head](image)

**Figure 5-3 Recovery Coordination Message vs. Number of Cluster Head**

In Figure 5-3 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, recovery probability by considering selection of CH and GW based on trust model and considering selection of CH and GW based on individual factors like failure rate, available battery power, availability and recommendation are compared. Recovery probability vs. varying individual factor while keeping the other three factors constant is also calculated and plotted later.

**Case1**: Selection of CH and GW based on failure rate.
Case 2: Selection of CH and GW based on availability of CM.

Case 3: Selection of CH and GW based on available battery power of CM.

Case 4: Selection of CH and GW based on recommendation from neighbor CM.

Case 5: Selection of CH and GW based on proposed Trust Model.

Figure 5-4. (a) Failure Rate of Node vs. Trust Value; (b) Available Battery Power of Node vs. Trust Value; (c) Recommendation from Neighbor Node vs. Trust Value

In Figure 5-4, relation of trust value with different factors is shown. In (a) if failure rate of node increases trust value will decrease. So, trust value is inversely proportional to failure rate. In (b) if availability of battery power increases trust value will increase. So, trust value is directly proportional to it. In (c) if recommendation from neighbor node increases trust value will increase. So, trust value is directly proportional to recommendation.

Figure 5-5. (a) Availability of Node vs. Trust Value; (b) Trust Value vs. Recovery Probability

In Figure 5-5, trust value vs. different factors of nodes are plotted. In (a) if availability of MH increases trust value will increase. So, trust value of MH is directly
proportional to availability. In (b), recovery probability increases with trust value. Recovery probability can be achieved as high as 0.99. In all the above analysis, failure rate is kept constant at 0.08.

![Graphs showing recovery probability vs. failure rate and percentage of availability vs. recovery probability.](image)

**Figure 5-6.** (a) Failure Rate vs. Recovery Probability Considering case1 and case5; (b) Percentage of Availability vs. Recovery Probability Considering case2 and case5

In Figure 5-6 comparisons of recovery probability vs. failure rate in different cases considered above are shown. In (a), if failure rate of CH and GW increases, 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 recovery probability of case5 and case1 are compared and found that recovery probability of case 5 is more than case1. In (b), if percentage of availability of CH and GW is increased, recovery probability will increase because failure rate will become low. As percentage of availability is inversely proportional to failure rate, percentage of availability is directly proportional to recovery probability. In above figure comparison between case5 and case2 shows that recovery probability of case5 is more than case2.
Figure 5-7. (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 5-7 recovery probability vs. different factors in different cases mentioned above are shown. In (a), 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. As available battery power is inversely proportional to failure rate, available battery power is directly proportional to recovery probability. In (a), comparison between case5 and case3 reflects the fact that recovery probability in case5 is more than case3.

In (b), if percentage of recommendation of CH and GW is increased, recovery probability will increase because failure rate will be low then. As percentage of recommendation is inversely proportional to failure rate, percentage of recommendation is directly proportional to recovery probability. In (b), comparing between case5 and case4 it is found that recovery probability of case5 is more than case4.

After analyzing the above comparisons following results are obtained.

<table>
<thead>
<tr>
<th>Desired recovery probability</th>
<th>Case1</th>
<th>Case2</th>
<th>Case3</th>
<th>Case4</th>
<th>Case5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.75</td>
<td>0.81</td>
<td>0.78</td>
<td>0.82</td>
<td>0.99</td>
</tr>
</tbody>
</table>
After analyzing Table 5.4, highest recovery probability can be obtained 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. In case5, CH and GW are selected based on above four factors together, so the chances of failure of CH and GW becomes lower. So maximum recovery probability using proposed trust model can be achieved is 99%.

Here a mobility based checkpointing and trust based rollback recovery algorithm combined with message logging is proposed 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. What makes more challenging the task of checkpointing in MANET is lack of stable storage. In MANET, checkpoint and log placement is also a vital issue. Checkpoint traversal through trusted mobile hosts only ensures successful recovery of failed mobile host. Failure prone components are also security attack prone. 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. The trust model described next has been extended to accommodate security.

5.3 Trusted Checkpointing

Existing secure checkpointing algorithm using cryptography to provide fault tolerance in failure and security attack prone mobile cellular network cannot be applied in MANET due to enhanced overhead. Moreover, MANET suffers from different types of security attacks [75][76]. A trusted MH will not be a malicious or selfish node, has low or negligible failure and security attack rate, high available battery power which indicates high availability in the network and high positive reference from other MHs. So checkpoint is secure in a trusted mobile host without additional overheads of cryptography.

A trust model is proposed using the basic concept of Ant Colony Optimization (ACO) [77][78]. The technique is based on the foraging mechanism employed by real ants attempting to find a shortest path from their nest to a food source. While foraging, the ants communicate indirectly via pheromone, which they use to mark their respective paths and which attracts other ants. It is a very popular technique used in MANET
because foraging technique and MANET have some similar characteristics like i) while searching food, ants move simultaneously, independently and without a supervisor. Similarly mobile hosts move randomly in the network and transfer packets without supervision of any centralized controller or supervisor and ii) each ant chooses a path on the basis of the pheromone deposit laid by other ants which have previously followed the same path. This cooperative behavior provides a positive feedback. This is similar to routing characteristics in MANET. In this technique, a dummy packet is initially forwarded by every MH in cluster based MANET before cluster head and gateway nodes are selected. Checkpointing is movement based where checkpoint will be taken and saved based on a predefined cluster change threshold.

Above mentioned literature survey helps to identify the following problems that motivated us to propose the work. In [43], checkpoints are always saved in cluster head. But maximum number of cluster members of a single cluster is not specified. In extreme case, if all cluster members want to save checkpoints at a time in cluster head, it may not be able to save all checkpoints due to limited memory space. In [71] trusted MH is selected on basis of only remaining energy and in [79] cluster head is selected based on trust value and less energy consumption. But there are many other factors like failure rate, availability in the network and security of node which may be low due to which the node or its task may fail. From above findings it is evident that existing works evaluate a MH based on a single factor to select cluster head. But practically this does not serve the purpose because though a node with lowest failure rate is selected as cluster head but its availability may be low due to link failure or shortage of battery power and suddenly disconnect from network. So this MH is not suitable as a cluster head or gateway. Hence there is a scope to combine multiple factors to select trusted cluster head and gateway.

5.3.1.1 System Model and Assumptions

System model used here is same as described in section 5.1.1. Along with assumptions mentioned in 5.1.1, following assumptions are made in addition.

i) Security of a checkpoint is defined by its integrity. Security attack to a checkpoint is considered in MHs, not in links
ii) All the MHs, on the selected shortest route between ‘cluster head that saves backup checkpoint’ and ‘cluster head in which failed MH recovers’, will not fail and will not be malicious during recovery process.

iii) Once a MH is evaluated as trusted, it will remain trusted throughout recovery process.

5.3.2 Trusted Recovery Based on ACO

Here the purpose is to ensure secure routing of checkpoints through trusted nodes only at the time of recovery. This can be fulfilled if following objectives are met: i) To find a path, from checkpoint to recovery MH, consisting of only trusted MHs and ii) If multiple paths exist, the path with minimum number of MHs and higher pheromone concentration value will be selected. Based on Ant Colony Optimization (ACO), such a path can be found. Pheromone deposit of a trusted MH is positive and these MHs are included in pheromone list which is maintained per node in the network. For checkpoint forwarding a node will select next node from its pheromone list.

Cluster based hierarchical MANET is considered here. Cluster member (CM) will save copy of its checkpoint in current cluster head (CH) after CM’s inter-cluster movement exceeds a predefined threshold [80] as described in the earlier sections of this chapter. Let suppose after moving through different clusters the CM fails. Now the CM will recover in a CH which will communicate through other CHs to find failed CM’s last saved checkpoint which will traverse through a number of CHs and GWs to reach the CH in which failed CM will recover. Using proposed trust model, all the nodes are evaluated to ensure that only trusted nodes are selected as CH and GW so that checkpoint traverses through only trusted nodes to implement secure routing of checkpoint [5].

- Formation of cluster

Each cluster has a cluster head and one or multiple gateway nodes through which it can connect with other clusters.
• **Cluster formation algorithm**

Each cluster consists of a predefined fixed number of nodes. If number of MHs in a cluster is less than this predefined number, then i) CH of this cluster periodically broadcasts a request message to its neighbor MHs to join, ii) A MH rapidly seeks to join a feasible cluster based on the advertisements from the neighboring CHs and iii) MH itself becomes a CH and starts broadcasting periodic request messages to its neighbor MHs for joining this newly formed cluster.

• **Cluster Head selection algorithm**

i) In a cluster, the CM with highest trust value will be elected as CH, ii) If more than one CM has same trust value, the CM with lowest failure rate will be considered as CH and iii) If more than one CM has same trust value and failure rate, then the CM having highest energy will be elected as CH.

• **Proposed Trust Model**

Trustworthiness of CM in a cluster is a belief that the CM will be secure, available, failure free and have sufficient energy such that its disconnection will be less. If it is impossible to decide whether a CM could be categorized in either ‘belief’ or ‘disbelief’ due to lack of communication or improper knowledge, it will be considered ‘uncertain’. Probability of a MH being trusted, distrusted and uncertain node can be expressed as given in [68] –

\[ b + d + u = 1 \]  

\[ b + d = 1 \]

Where \( b= \) belief, \( d= \) disbelief and \( u= \) uncertain. Here belief is considered as trusted, disbelief as distrusted but status of any node cannot be uncertain. So, above equation 6 can be re-written as

\[ b + d = 1 \]

There are various methods of trust value evaluation of a system in [4][68] and removal of uncertainty in [46]. In proposed algorithm, trust value of a node is defined using a trust_table which consists values of three factors: successful response rate, failure rate and percentage of available battery power.
• **Successful response**

It means when a MH communicates with another MH and both respond successfully, then it is considered that both MHs are available in the system. When any MH is defined as trusted it must be secure. Successful response may not indicate whether MH is selfish or malicious. Concept of Ant Colony Optimization is used here for this purpose. After getting successful response, each MH calculates hash of a dummy packet and sends it to all the MHs connected to it. The MHs that receive it, calculate hash and send back to sender MH attached with an acknowledgment. If two hashes match at sender mobile host, then corresponding receiver MH is not malicious. Then pheromone deposit on the MH is considered to increase otherwise decreases. From source MH to destination MH there may be multiple MHs that give successful response, checkpoint will traverse through the path that has less number of MHs and higher pheromone deposit.

• **Algorithm to spread pheromone**

Before describing algorithm, some of the notations used are illustrated first. 

\(CMS = \) sender cluster member, \(CM_R = \) receiver cluster member, \(D_{pkt} = \) dummy packet that traverse through MHs as ‘ant’, \(ACK_{pkt} = \) acknowledgement packet, \(S_r = \) successful response rate, \(P_c = \) pheromone concentration, \(S_{r\_count} = \) successful response rate counter, \(T_{val} = \) trust value, \(T_{val\_th} = \) threshold of trust value.

1. CMS calculates hash of \(D_{pkt}\), appends ID of all CM_R at 1-hop distance with CM_S and ID of destination node to \(D_{pkt}\) and forwards to CM_Rs. So, each CM_R will get to know the IDs of all other CM_Rs.

2. CM_R receives \(D_{pkt}\): CM_Rs calculate hash, attach to \(ACK_{pkt}\) and send back to CM_S. CM_S receives \(ACK_{pkt}\), de-attaches hash and compares with previously saved hash.

   2.1 if a match is found, CM_S sets \(S_r = 1\), increments \(S_{r\_count}\) by 1 and includes corresponding CM_R – id in its pheromone list. Then \(P_c\) is calculated using equation 8.

   2.2 else CM_S sets \(S_r = 0\), decrements \(S_{r\_count}\) by 1, marks the MH as malicious and does not include corresponding CM_R in pheromone list.

   2.3 else CM_R send back \(ACK_{pkt}\) to CM_S within a stipulated time,
then CMS marks it as ‘selfish node’ (which may have simply dropped the packet to save its resources as the packet is not of its interest)

3. CM_R checks the receiver list saved in D_pkt. If any of the CM_R-ID from this list exists in current CM_R’s 1-hop distance neighbour list, then current CM_R will not forward D_pkt to the common nodes that have already received D_pkt but forwards to others.

4. Repeat steps 2-3 till destination node is reached.

• Pheromone calculation

\[ P_c = ( S_r \times (S_{r\text{.count}} + 1)) + ((1 - S_r) \times (S_{r\text{.count}} - 1)) \]  

(8)

The amount of pheromone that is deposited depends on the parameter S_r i.e. successful response between two nodes.

This concept is illustrated with the help of following figure:

![Figure 5-8 Example of Spread of Pheromone in Cluster](image)

In above figure, there are two paths from A to D: P_1 = A - B - D, P_2 = A - B - C - D. First A will forward a dummy packet to destination MH i.e. D to find a trusted path. A calculates hash of a dummy packet and saves in its memory, appends destination MH-ID and list of its immediate neighbor i.e. B to dummy packet and sends to its immediate neighbor i.e. B. B receives dummy packet and checks destination MH-ID which does not matches with its own ID. B then sends back an acknowledgement with hash of dummy packet to CM_S i.e. A. Then A will perform hash comparison between saved and received hashes. Following cases may happen:

**CASE 1:** if a match is found then A gets successful response from B. Now A will set S_r = 1, increment S_{r\text{.count}}, calculate P_c of B according to Equation 3 and add B in its pheromone list.

**CASE 1.1:** B attaches its immediate neighbors’ list, CM_S-ID, destination MH-ID to dummy packet and forwards to C, D. B gets successful response from all the MHs and
sends back to CMs. D finds a match between destination node-ID and own ID. So along with successful response it sends a hello message ‘I am the destination’ to CMs i.e. to A through B. B adds A, C, D in its pheromone list and A adds C and D in its pheromone list. C does not find a match between own-ID and destination MH-ID. So it continues with dummy packet forwarding and adds D, B in its pheromone list on getting successful response from these MHs. As stated before, D again sends a hello message ‘I am the destination’ to CMS i.e. to A through C and B. Now A gets two pheromone paths, P1: A-B-D and P2: A-B-C-D. P1 consists of less number of MHs, hence it is selected to forward checkpoint from A to D.

**CASE 1.1.1:** D gives successful response to A through B. B and A both add D in their corresponding pheromone list same as stated above. But C does not respond to A i.e. Sr of C is 0. Both A and B term C as ‘malicious’ and exclude C from pheromone list. So P2 will not be selected as pheromone path.

**CASE 2:** if a match is not found then A does not get successful response from B, terms B as ‘malicious’, does not include in pheromone list and does not further receive or send any packet from or to B respectively. Hence checkpoint cannot be forwarded to the destination node.

This ACO Based trust evaluation of MHs helps to find following three factors:

**Availability of MH:** It can be determined if a MH is currently connected to the network or not. A MH is available if it returns an acknowledgement packet within a stipulated time otherwise the MH is unavailable.

**Data integrity on a MH:** This is very important characteristic because checkpoint is saved in or transferred through CH or GWs. Hence data integrity of checkpoint must be maintained on all MHs. Integrity of checkpoint is important for successful recovery of failed MH from its last checkpoint.

**Selfish behavior of MH:** It means a mobile host will not forward packet for saving its resources intentionally. As a checkpoint is transferred through different MHs to reach to the failed MH, it is necessary to check that any intermediate node is not ‘selfish’ [14].

If a MH is available in the network and successfully forwards checkpoint keeping its integrity then the MH gives successful response i.e. Sr = 1. Along with this, following two factors are considered to find a MH is trusted or not.
Failure rate ($F_r$): A trusted MH must have low failure rate so that it can forward checkpoint without fail.

Percentage of battery power left ($\%B_{left}$): A mobile host having sufficient energy can forward checkpoint without having any possibility of deliberate switch off due to lack of battery power.

Range of values of these two factors is considered from 0 to 1. Threshold is considered to be 0.5. Desired range of failure rate is 0 – 0.5 and $\%B_{left}$ is 0.5 – 1. Based on the values of these factors, one MH can set a preference value ($P_{pref}$) for other MH.

- **Personal preference ($P_{pref}$) value list of each MH**

<table>
<thead>
<tr>
<th>Range</th>
<th>$F_r$</th>
<th>$%B_{left}$</th>
<th>$S_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&gt; 0.5</td>
<td>= 0.5</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>$P_{pref}$</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Trust value calculation:**

$$T_{val} = (F_r \times P_{pref}) + (%B_{left} \times P_{pref}) + (S_r \times P_{pref}) \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots (9)$$

Threshold of different factors are: upper threshold of $F_r$ = 0.5 and lower threshold of $\%B_{left}$ = 0.5 as low $F_r$ and high $\%B_{left}$ is desired. So a trusted node must have $F_r < 0.5$ and $\%B_{left} > 0.5$ and $S_r$ must be 1. So threshold of trust value, $T_{val,th} = \{(0.5 \times 0.5) + (0.5 \times 0.5) + (1 \times 1)\} = 1.5$. These values are taken from Table 5.5. A MH is trusted if its $T_{val} > 1.5$.

- **Algorithm to compute trust value:**

1. Each CM will maintain own trust_table that contains above three factors and keeps trust_tables of other CM of same cluster.
2. Then each CM sets $P_{ref}$ for the above three factors for each of its neighbor CM and evaluates $T_{val}$ using equation 4.
3. if ($S_r = 1$)
   
   \{ if ($T_{val} > T_{val,th}$) \&\& ($P_c \geq 1$) \&\& ($F_r < 0.5$) \&\& ($\%B_{left} > 0.5$) \}, the MH is trusted;
else the MH is distrusted;
else if (Sr = 0), the MH does not give successful response, MH is distrusted;
4. Now each CM knows which CM has highest Tval. This helps to elect a CM as CH of a cluster.
5. Once each MH gets trust_table of other MHs, selection of cluster head has been done, then each MH deletes trust_table of other MHs, transfers own trust_table to current cluster head. Later changes of trust factors of each MH are intimated to cluster head only.

Table 5.6 shows trust_table of 4 MHs of a cluster. Based on Table 5.5 and Table 5.6, trust values of these 4 MHs are calculated and justification of Pref values of each factor can also be understood. Here values of different factors are taken arbitrarily within a range from 0 to 1 for illustration purposes.

<table>
<thead>
<tr>
<th>MH</th>
<th>Fr</th>
<th>%Bleft</th>
<th>Sr</th>
<th>Pref</th>
<th>Trust value</th>
<th>Level of MH</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH1</td>
<td>0.001</td>
<td>0.6</td>
<td>1</td>
<td>(1,1,1)</td>
<td>1.601 &gt; 1.5</td>
<td>Trusted</td>
</tr>
<tr>
<td>MH2</td>
<td>0.03</td>
<td>0.5</td>
<td>1</td>
<td>(1,5,1)</td>
<td>1.28 &lt; 1.5</td>
<td>Distrusted</td>
</tr>
<tr>
<td>MH3</td>
<td>0.4</td>
<td>0.8</td>
<td>1</td>
<td>(1,1,1)</td>
<td>2.2 &gt; 1.5</td>
<td>Trusted</td>
</tr>
<tr>
<td>MH4</td>
<td>0.0001</td>
<td>0.8</td>
<td>0</td>
<td>(1,1,0)</td>
<td>0.8001 &lt; 1.5</td>
<td>Distrusted</td>
</tr>
</tbody>
</table>

MH1: Sr = 1, %Bleft = 0.6, so according to Table 5.6, Pref = 1, Fr = 0.001, Pref = 1. Now, Tval of MH1 = ((0.001*1) + (0.6*1) + (1*1)) = 1.601 > 1.5. So Tval of MH1 is > Tval_th. So MH1 is trusted.

Suppose, for Sr = 1, Pref < 1, say 0.8. Then Tval = ((0.001*1) + (0.6*1) + (1*0.8)) = 1.401 < 1.5 which indicates MH1 as distrusted although Fr < 0.5 and %Bleft > 0.5 which are sufficient to consider a node as trusted. Thus, Pref set at 1 when Sr = 1 is justified. Similarly, other Pref values also can be justified.

MH2: Sr = 1, %Bleft = 0.5, so according to Table 5.6, Pref = 0.5, Fr = 0.03 < 0.5, Pref = 1. Tval of MH2 = ((0.03*1) + (0.5*0.5) + (1*1)) = 1.28 < 1.5. So Tval of MH2 is < Tval_th. Hence MH2 is distrusted which is supported by the value of %Bleft which must be > 0.5 in
a MH to be trusted. MH2 have %B\_left = 0.5 which is not sufficient for a MH to be trusted and this is supported by trust value also.

**MH3:** Sr = 1, %B\_left = 0.8, P\_pref = 0.1, F_r = 0.4 < 0.5, so according to Table 5.6, P\_pref = 1.

\[ T_{val\ of\ MH3} = ((0.4*1) + (0.8*1) + (1*1)) = 2.2 > 1.5. \]

So T\_val of MH3 is > T\_val\_th. Hence MH3 is trusted.

**MH4 :** Sr = 0. So the MH is ‘malicious’ and considered as distrusted irrespective of its low F_r and high %B\_left.

As MH1, MH2, MH3 are member nodes of a single cluster then after above evaluation, MH1 adds MH3 in its pheromone list and MH3 adds MH1 in its pheromone list and MH3 with highest trust value is elected as cluster head.

### 5.3.2.1 Working Example

![Figure 5-9](image)

**Figure 5-9 (a) Working Example of the Network Considered in our Work; (b) Formation of Cluster after Evaluating Trust Value of MHs**

Cluster based MANET considered here is shown in Figure 5-9 (a). Election of CH of cluster C\_1 based on trust value is explained below. CH election of C\_2, C\_3 will be similar to that of C\_1. Each MH will calculate successful response of other MHs through sending dummy packet and calculating pheromone concentration and list.

- **Election of cluster head and GW**

Each MH broadcasts IDs of elected CH and GW. Thus CH and GW can be elected based on their Trust value. Here for example it is described how B computes Trust value of own and other MHs as shown in Table 5.7. Similarly, other nodes also individually calculate trust value of own and other nodes within same cluster.
Table 5.7 Trust Values of all CMs in C\textsubscript{1} as Shown in Figure 5-9

<table>
<thead>
<tr>
<th>Mhs</th>
<th>F\textsubscript{r}</th>
<th>%B\textsubscript{inf}</th>
<th>S\textsubscript{r}</th>
<th>T\textsubscript{val}</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.04</td>
<td>0.5</td>
<td>1</td>
<td>1.29</td>
<td>Distrusted</td>
</tr>
<tr>
<td>B</td>
<td>0.03</td>
<td>0.65</td>
<td>1</td>
<td>1.68</td>
<td>Trusted, CM</td>
</tr>
<tr>
<td>C</td>
<td>0.04</td>
<td>0.8</td>
<td>1</td>
<td>1.84</td>
<td>Trusted, CH</td>
</tr>
<tr>
<td>D</td>
<td>0.001</td>
<td>0.5</td>
<td>0</td>
<td>0.251</td>
<td>Malicious, distrusted</td>
</tr>
<tr>
<td>E</td>
<td>0.01</td>
<td>0.7</td>
<td>1</td>
<td>1.71</td>
<td>Trusted, GW</td>
</tr>
<tr>
<td>F</td>
<td>0.03</td>
<td>0.55</td>
<td>1</td>
<td>1.58</td>
<td>Trusted, CM</td>
</tr>
</tbody>
</table>

In above table, C’s T\textsubscript{val} is highest, hence will be selected as CH. Next among B, E and F, E’s T\textsubscript{val} is highest. So E will be selected as GW. Suppose CH of C\textsubscript{2} is H and GW is M and in C\textsubscript{3} CH is K. After election of CH and GW, cluster will be like as shown in Figure 5-9(b).

All the nodes in the network shown in Figure 5-9 (a) are evaluated using proposed trust model. A is found to be distrusted and D malicious as shown in Figure 5-9 (b). So in cluster C\textsubscript{1} only B can be CH if C leaves C\textsubscript{1} or its trust value falls below 1.5 over time.

- **Checkpointing and recovery**

CMs take checkpoint based on inter-cluster movement [80]. Suppose D will take checkpoint when c3\_th=2. So when D will reach C\textsubscript{3} after visiting C\textsubscript{2}, it will take checkpoint. D’s last checkpoint is saved in C. After saving checkpoint two cases may happen.

**Case1:** D fails in C\textsubscript{3} before saving checkpoint. Failure notice goes to K (CH of C\textsubscript{3}) and later D will recover in K. K broadcasts recovery-notice of D to all CHs. C finds a match with D’s ID from saved checkpoint-list and forwards D’s last checkpoint to K through C-E-H-M-K path. K sends checkpoint to D. D then rolls-back and recovers from failure.

**Case2:** D saves checkpoint successfully in K and sends request to C to delete checkpoint. After receiving message, D will delete checkpoint.
5.3.3 Performance Analysis

- Recovery Probability

\[ P_{\text{Rec}} = (1 - \frac{\text{number of distrusted nodes}}{\text{total number of MHs}}) \]  

The simulation environment is same as that mentioned in section 5.2.4. The parameters which have been used in our simulation are represented below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value or range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkpoint size</td>
<td>256 kb</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>20</td>
</tr>
<tr>
<td>Max. number of CM a cluster can support</td>
<td>5</td>
</tr>
<tr>
<td>Pheromone concentration</td>
<td>10-100%</td>
</tr>
<tr>
<td>Channel bandwidth</td>
<td>1MB</td>
</tr>
<tr>
<td>Mobility rate</td>
<td>0.4 and 0.48</td>
</tr>
<tr>
<td>Failure rate</td>
<td>0.02-0.08</td>
</tr>
<tr>
<td>Cluster change threshold</td>
<td>3-20</td>
</tr>
</tbody>
</table>

- Experimental Result

Following three cases are considered based on selection of CH and GW selection procedure and compare the recovery probability among them.

Case 1: Selection of CH and GW based on failure rate only.

Case 2: Selection of CH and GW based on \( %B_{\text{left}} \) of CM.

Case 3: Selection of CH and GW based on proposed Trust Model.

In Figure 5-10, trust value vs. failure rate, battery power and successful response are shown. In (a) if failure rate of a MH increases, its trust value will decrease. In (b) if \( %B_{\text{left}} \) increases, trust value will increase. In (c) if successful response rate increases, trust value will increase.
In (a), (%B\textsubscript{left}) is kept constant and its value is set at 0.5, successful response rate at 1. Trust value of MH increases up to failure rate = 0.4 because for a range of failure rate from 0.1 to 0.4, P\textsubscript{pref} value will be 1. For example, for failure rate 0.4 trust value will be (0.4*1) + (0.5*0.5) + (1*1) = 1.65. Then for failure rate 0.5 trust value will be (0.5*0.5) + (0.5*0.5) + (1*1) = 1.5. Then trust value for the range of failure rate from 0.6 to 1.0 will remain same i.e. 1.25 because, personal preference value for this range of failure rate will be 0. For example, for failure rate = 0.7, trust value will be (0.4*0) + (0.5*0.5) + (1*1) = 1.25. So, trust value is inversely proportional to failure rate.

In (b), Failure rate is kept constant at 0.5 and successful response rate is 1. T\textsubscript{val} starts increasing when (%B\textsubscript{left}) is > 0.5. because personal preference value(P\textsubscript{pref}) for this range of (%B\textsubscript{left}) is 1. For example for (%B\textsubscript{left}) 0.6, trust value will be (0.5*0.5) + (0.6*1) + (1*1) = 1.85. Then for (%B\textsubscript{left}) 0.5 trust value will be (0.5*0.5) + (0.5*0.5) + (1*1) = 1.5. Then for range of (%B\textsubscript{left}) between 0.1 - 0.4, trust value will remain same i.e. 1.25 because for personal preference value will be 0. For example, for failure rate 0.3, trust value will be (0.5*0.5) + (0.3*0) + (1*1) = 1.25. So, trust value is directly proportional to battery power.

In (c), 0 indicates unsuccessful response and 1 indicates successful response. Here B\textsubscript{left} is kept constant at 0.5 and failure rate at 0.5. So for successful response rate = 0, trust value will be (0.5*0.5) + (0.5*0.5) + (0*0) = 0.5 and for successful response rate = 1, trust value will be (0.5*0.5) + (0.5*0.5) + (1*1) = 1.5. So, T\textsubscript{val} is directly proportional to S\textsubscript{r}.
In Figure 5-11 selfish and malicious node vs. recovery probability are plotted. In (a) if number of selfish node increases recovery probability will decrease because selfish MH will not forward packet. So recovery probability will decrease. In (b) if number of malicious MH increases, recovery probability will decrease. Malicious MH will change checkpoint data. If checkpoint data is changed failed CM can not recover. So recovery probability will decrease.

Recovery probability of a failed MH with its checkpoint routed through a number MHs having varying % of pheromone deposit is shown in Figure 5-12.

Figure 5-11. (a) % of Selfish MH vs. Recovery Probability ; (b) % of Malicious MH vs. Recovery Probability

Figure 5-12. (a) % of Pheromone Concentration vs. Recovery Probability; (b) Comparison of Recovery Probability between case1, case3 and case2, case3
In (a) % of pheromone concentration increases, number of trusted MH also increases. So recovery probability will increase. Recovery probability of case1, case3 and case2, case3 are compared and following results are obtained as given in Table 5.9. This result is plotted in (b) of above figure.

**Table 5.9 Performance Analysis**

<table>
<thead>
<tr>
<th>desired recovery probability</th>
<th>Highest Recovery Probability from experimental result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case1</td>
<td>0.86</td>
</tr>
<tr>
<td>Case2</td>
<td>0.80</td>
</tr>
<tr>
<td>Case3</td>
<td>0.93</td>
</tr>
</tbody>
</table>

After analyzing above results, it is found that highest recovery probability is obtained in case3 which is very nearest to desired recovery probability of any fault tolerant system. For case1 CH and GW will be selected based on only failure rate. 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 low availability or security attack. So checkpoint may not be transferred to the destination cluster head all the times. This reason is also applicable for case2 as mentioned above. But in case3 CH and GW has been selected considering all the above mentioned factors. So probability of failure of CH and GW will be less. Thus 93% recovery probability can be achieved.

Although many trust models are proposed, failure rate of a mobile host is not yet considered in any of the existing trust model. Proposed trust model considers failure rate as a factor to decide if a mobile host is trusted or not. Here ant colony based trust model is proposed to evaluate MHs and to ensure secure routing of checkpoint only through trusted MHs without adding cryptography overheads. Combination of mobility based checkpointing and trust based recovery is unique in this category. Trusted checkpointing will find its application more and more in such systems with the advancement of technology and evolution of service oriented systems.

**5.4 Conclusion**

The trust models proposed here evaluate mobile hosts as trusted or not based on different parameters for example failure rate, security attack rate, battery power,
availability in the network etc. A trusted mobile host ensures successful communication and checkpoint transfer without any failure and provides security as explained earlier. Now, if any of the checkpoint forwarding mobile host between checkpoint and recovery node is distrusted, checkpoint transfer may fail or the checkpoint may suffer from security attack. Then again trusted checkpoint forwarding nodes between checkpoint and recovery node has to be discovered afresh. If this is repeated then it will cause delay in checkpoint transfer and this in turn will delay the recovery of failed mobile host. Moreover, recovery may be unsuccessful. For successful recovery, primary concern is integrity of checkpoint and to ensure that propose a technique using cryptography and trust is proposed in the following chapter.