The term outlier detection has different meaning in different contexts. For example in Data Mining, outlier detection means finding out data which is unusual to some extent. In MANETs, outlier detection deals with finding out those nodes which are of distinguished nature as compared to the regular nodes of the network. Such types of nodes are also called as misbehaving nodes. Hence outlier detection in MANETs, refers to finding out all types of misbehaviors observed in the network. The misbehavior in MANETs is termed as packet drop, misroute, false request or acknowledgements, packet modification etc. The cause of these misbehaviors may be due to some faults in the network like power failures or specific features of such type of networks e.g. dynamic topology, multi-hop connectivity etc. These misbehaving nodes are responsible to a large extent for degrading the performance of the network. For example, if a node launches an attack in the network by dropping all the packets, throughput of the network gets down. Such type of misbehaving nodes can be caught by outlier detection methods as the behaviour of those nodes would be different to other nodes of the network. So, the main objective of outlier detection in MANETs is to isolate those nodes which are misbehaving in the network despite of the sources of misbehavior.

In this chapter, an algorithm (OUTM) is proposed for identifying outlier nodes using trust management (Renu Popli, et al., 2018), which accomplishes the second objective of the present research work. The working of the proposed outlier detection algorithm is defined by means of a layered structure model. The layered structure model is divided into three layers. Each layer defines the respective part of the algorithm.

The organization of this chapter is as follows: Section 5.1 provides the introduction to the three layered model which describes various components of the proposed algorithm. Section 5.2 illustrates the proposed algorithm for outlier detection using trust management. Section 5.3 describes the implementation of proposed algorithm. In Section 5.4 simulation results and comparative analysis with existing outlier detection approaches SMART and mTrust are produced. Section 5.5 provides the outcome of
the chapter.

### 5.1 Introduction to Layered Model of Outlier Detection

The model describing various layers of outlier detection is shown in Figure 5.1. This model consists of three parts:

- **Part-I** Data Layer
- **Part-II** Implementation Layer or Trust Management Layer
- **Part-III** Application Layer

![Proposed Layered Model of Outlier Detection](Image Source:[Renu Popli, et al., 2017]

Figure 5.1: Proposed Layered Model of Outlier Detection

The model in Figure 5.1 defines three parts of the outlier detection implementation. Part-I specifies the data layer which deals with the collection of behaviour data for calculating trust parameters of the nodes. Part-II specifies the implementation layer or trust management layer which includes trust generation, propagation and updation. And Part-III specifies application layer which applies the value of trust for specific
purpose such as for outlier detection. A detailed description of these layers is given below:

5.1.1 Data Layer

This layer consists of Behavior Data Collection component, which deals with collecting behaviour information of the nodes and generating behavioral datasets. In the present research work, network simulation is used to produce datasets containing behavior data and KNN classifier is trained correspondingly.

At this layer, the behaviour data contains the data collected through direct as well as indirect observations from the nodes. The direct information is taken from the neighboring nodes which help in generating knowledge parameter of trust. The indirect information is collected from the nodes which are not directly linked but have some common nodes in between. This information assists in calculating recommendations about the common nodes. The data at data layer is further processed to generate trust parameters which are the output of this component.

In the present research work, each node observes its neighbors in order to collect information about their behavior. By observing behaviour of the nodes, the parameters like PMOR, PMIR and PDR are calculated as discussed in Section 4.2.1 of Chapter 4. By using these three parameters, a dataset is generated by differentiating malicious nodes from the normal nodes. A dataset of 50 nodes is given in Appendix-II. A specimen of dataset is shown as given below in Table 5.1.

<table>
<thead>
<tr>
<th>Node_id</th>
<th>Time</th>
<th>Is_CH</th>
<th>Is_Mal</th>
<th>Dis_to_CH</th>
<th>PDR</th>
<th>PMIR</th>
<th>PMOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>17.03</td>
<td>0.69</td>
<td>0.73</td>
<td>0.61</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16.65</td>
<td>0.54</td>
<td>0.27</td>
<td>0.84</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>17.58</td>
<td>0.94</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16.21</td>
<td>0.58</td>
<td>0.35</td>
<td>0.28</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

So, at this layer each node generates local behaviour table of its neighboring nodes. Output of behaviour data collection component is passed to the implementation layer to generate trust tables.
5.1.2 Implementation Layer

This layer is also called Trust Management Layer as (Renu Popli, et al., 2017) it deals with management of trust in the network. At this layer, each node generates trust values of its neighboring nodes by using three parameters of trust which is called as direct trust. In the present research work, the trust value is an aggregate of three terms which are knowledge, experience and recommendations. These are considered as important dimensions of calculating trust of a node. The details of generating trust value using these three dimensions of trust is given in trust model in Section 4.2.1 of Chapter 4. At this layer, Trust Management consists of Trust Generation, Trust Propagation and Trust Updation, which are explained as below:

- **Trust Generation**

Trust generation involves generation of the following tables:

- Local Trust Table (LTT)
- Global Trust Table (GTT)

Every node in MANETs calculates its local trust table which consists of trust value of their neighboring nodes. These trust values are calculated by using Equations 4.1 and 4.2 as discussed in Section 4.2 of Chapter 4, which consists of three different dimensions of trust which are knowledge, experience and recommendations.

An example of generating LTT is defined by taking a set of nodes as given below in Table 5.2.

<table>
<thead>
<tr>
<th>Input</th>
<th>Trust Calculation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of nodes – 6 (A,B,C,D,E,F), A_E_B, A_K_B, A_R_B, W_E=0.25, W_K=0.5, W_R=0.25</td>
<td>_A_T_B = W_E * _A_E_B + W_K * _A_K_B + W_R * _A_R_B</td>
<td>LTT</td>
</tr>
</tbody>
</table>

*Table 5.2: Generation of LTT*
In Table 5.2, for given 6 number of nodes, weights values are defined in the table. By using the trust formula shown in table, LTT is generated. Let A consists of 3 neighbor nodes B, C and D then LTT of node A consists of trust values of B, C and D. The trust value of A to B is calculated as:

$$A_TB = (0.2 \times 0.25) + (0.6 \times 0.5) + (0.2 \times 0.25)$$

$$= 0.4$$

Where, the values of weights are taken from Table 5.2 and the corresponding values of $A_{EB}$, $A_{KB}$ and $A_{RB}$ are taken from Table 5.3. Similarly other values of trust are calculated and the LTT generated for node A is shown in Table 5.3 below:

**Table 5.3: Local Trust Table**

<table>
<thead>
<tr>
<th></th>
<th>$A_{EB}$</th>
<th>$A_{KB}$</th>
<th>$A_{RB}$</th>
<th>$A_{TB}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>C</td>
<td>0.7</td>
<td>0.3</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td>D</td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

- **Trust Propagation**

Once all the nodes generate their local trust tables, they exchange it with their neighboring nodes. Trust tables go on propagated in the network whenever there are updations in the network.

![Figure 5.2: Trust Propagation](image-url)
This is required because of obtaining the up to date version of trust tables. The process of trust propagation continues till Global Trust Tables are generated.

The trust propagation is explained with the help of an example is shown in Figure 5.2, node A exchanges its trust table with its neighbours B, C and D as they are its neighbors. Similarly, node B exchanges its trust table with C, A and F and so on. The values shown in the trust tables corresponding to each nodes are assumed for explanation.

- **Trust Updation**

After receiving Local Trust Tables from neighboring nodes, each node updates its LTT based on the received table. In the algorithm proposed in upcoming Section 5.3, the simple average method is used for updation of trust tables. In this method, if a node is common between two trust tables, then the trust value of that node is calculated by taking average of the values from both tables. All the other nodes which are distinct in the table are added to the updated table with their calculated trust values. As a result, an updated trust table is generated.

The process of trust table updation is described through example as shown in Figure 5.3.

![Figure 5.3: Trust Updation](image)
In Figure 5.3, node A receives three trust tables from its neighboring nodes and updates its own trust table by applying simple average method of updation. The updated table at node A contains the trust values of all the nodes in the network.

5.1.3 Application Layer

This layer deals with the application of trust for some specific purpose. In the proposed model, this layer deals with the detection of outlier nodes from the network. The Local Trust Tables in the trust updation phase continues to update till Global Trust Tables are generated. Nodes in the GTT are arranged in increasing order of their trust values. The node which lies on the top position in GTT is least trustworthy nodes. So the top k nodes are chosen to be outliers as compared to the remaining nodes. The algorithm for outlier detection using trust management is given in the next section.

5.2 Proposed Algorithm1: Outlier Detection Algorithm using Trust Management (OUTM)

The aim of the proposed algorithm (Renu popli, et al., 2018) is to filter out k number of outlier nodes which show misbehaviors such as dropping of packets, modification of packets or misrouting of packets. The variable k has some integer value and is defined by the user.

Here, the network is assumed to contain a N number of mobile nodes. The size of the network may change according to connecting or disconnecting of the nodes. However, some failures of battery power exhaustion etc. may also reduce the size of the network. The outlier detection algorithm is described below to find out outliers.

Algorithm1: Outlier Detection Algorithm in MANETs (OUTM)

Input: LTT (Local Trust Table), N (total number of nodes)

Output: GTT\(_K\) (Global Trust Table containing K number of outlier nodes)

Method:

Begin

For every node \(n \in N\):

Exchange LTT, with its all direct neighbors


Upon receiving $LTT_i$ from node $n_j$:

**Call** $T_{\text{Merge}}(LTT_i, LTT_j)$;

When no further data switching performs:

$\forall k, GTT_k = LTT_k$

**End**

//TRUST UPDATION ALGORITHM

**Procedure** $T_{\text{Merge}}(LTT_i, LTT_j)$

**Begin**

On receiving $LTT_j$ from node $n_j$:

**If** $LTT_j \neq LTT_i$

Merge $LTT_i$ and $LTT_j$ based on following cases:

**Case 1**: If node $p$ occurs in both $LTT_i$ and $LTT_j$,

then take average of the corresponding columns for
node $p$ in both tables and store the average of node $p$ to
$LTT_i$.

**Case 2**: If node $p$ is in either of $LTT_i$ or $LTT_j$,

then add an entry of $p$ to the table $LTT_i$.

Broadcast $LTT_i$ to all of its direct neighbors

**else**

Keep $LTT_i$ unchanged and do not broadcast any message.

**return** $LTT_i$.

**End**

5.3 Implementation

The proposed algorithm is implemented by coding done in MATLab. The code for the OUTM algorithm is given in Appendix-IV. During implementation, 50 number of
nodes are taken and the value of $k$ is set to 5. The snapshot of the output obtained through simulation is shown in Figure 5.4.
In Figure 5.4, topkMalicious variable generate top k no. of nodes from the global trust matrix output. The variable mi denotes the malicious nodes introduced in the algorithm and int variable generates the intersection of the topKMalicious and mi. In the above figure the result is 100% accurate. It means that the OUTM algorithm is capable of finding out the malicious nodes from the network. However, the simulation is performed for 10 different rounds and average of the results is calculated and then performance is evaluated by using three criteria which are given below:

- Precision: it is the fraction of retrieved data that is relevant to the query. For example in MANETs, it gives the ratio of no. of truly outlier nodes detected to the total no. of untrustworthy nodes found.

- Recall: it is the fraction of the data that is relevant to the query that is successfully retrieved. For example in MANETs, it no. of truly outlier nodes detected to the total no. of truly outlier nodes.

- Fmeasure: it is a balanced score that combines precision and recall by harmonic mean.

Figure 5.4: Output of OUTM Algorithm
These parameters are calculated as follows:

\[
\text{Precision} = \frac{\text{Relevant Data}}{\text{Retrieved Data}}
\]

\[
\text{Recall} = \frac{\text{Retrieved Data}}{\text{Relevant Data}}
\]

\[
F\text{measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

The Figure 5.5 illustrates the results obtained after calculating the above three parameters.

5.4 Simulation Results And Comparative Analysis

The proposed algorithm is simulated by using MATLAB R2012a simulator. The proposed algorithm is compared with two existing outlier detection algorithms mTrust and SMART and results are produced.

mTrust (Wenjia Li, et al., 2010) is a trust management framework which uses three directions of evaluating trust which are collaboration, behavioral and reference trust. This framework is used to identify misbehaving nodes in MANETs. However,
accuracy of detecting misbehaviors is reduced when node density as well as mobility increases.

SMART (Wenjia li, et al., 2010) is also a trust management framework which makes use of SVM (Support Vector Machine) approach to classify nodes into misbehaving nodes and regular nodes. The performance of this approach is better than mTrust in some scenarios when every time attack patterns are not the same. However, the performance of SMART gets reduced when percent of misbehaving nodes increases.

The performance is measured by varying simulation parameters. The simulation results are presented by taking three scenarios as defined in Table 3.1 of Chapter 3.

Table 5.4: Simulation Scenarios for OUTM

<table>
<thead>
<tr>
<th>Simulation Parameters</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Area</td>
<td>100x100</td>
<td>100x100</td>
<td>100x100</td>
</tr>
<tr>
<td>Total Number of nodes</td>
<td>50</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Transmission Range</td>
<td>60</td>
<td>90</td>
<td>120</td>
</tr>
<tr>
<td>Simulation Time</td>
<td>10 units</td>
<td>10 units</td>
<td>10 units</td>
</tr>
<tr>
<td>Mobility</td>
<td>5 mtr/sec</td>
<td>10 mtr/sec</td>
<td>20 mtr/sec</td>
</tr>
<tr>
<td>Number of Malicious Nodes</td>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

As defined in Table 5.4, the performance of three algorithms to be compared, are measured by varying number of nodes, mobility, transmission range, and number of malicious nodes. The values of precision and recall are calculated for the same and shown in Figures 5.6 to Figure 5.13.
Figure 5.6 shows the effect of misbehavior over precision. With increase in the number of malicious nodes, the precision of mTrust and SMART gets reduced but the proposed algorithm (OUTM) shows slight change in precision as compared to both approaches.

Figure 5.7 shows the effect of mobility over precision value. The proposed algorithm is suitable for less dynamic environment. So, as the mobility of the nodes increases, precision of mTrust, SMART and OUTM get reduced. However, SMART and OUTM shows similarity in their behavior.
Figure 5.8: Precision v/s No. of Nodes in OUTM

Figure 5.8 shows the effect of node density over precision value. As no. of nodes increases, the precision value should increase. This occurs because the ratio of trustworthy nodes over misbehaving nodes gets increased, as range of misbehaving nodes is fixed for this scenario in the proposed algorithm. So the graph shows an improvement in the precision value for the proposed algorithm as compared to both approaches.

Figure 5.9: Precision v/s Transmission Range in OUTM

Figure 5.9 shows the effect of varying transmission ranges over precision value. As transmission range increases, no. of neighbors in the list increases which results into
more collection of recommendations in our proposed work hence shows an increase in precision value.

Figure 5.10: Recall v/s Malicious Nodes in OUTM

Figure 5.10 shows the effect of varying misbehaving nodes over recall value. With increase in percentage of misbehaving nodes, the recall value gets decreased for both approaches mTrust and SMART. However, the proposed algorithm shows very little effect and has higher recall values as compared to both approaches.

Figure 5.11: Recall v/s Mobility in OUTM

Figure 5.11 shows the effect of mobility over recall values. As mobility increases, Recall of mTrust, SMART and OUTM get reduced. This is true because it will be
more challenging for all the three to detect the true outlier nodes when mobility increases. However, SMART and OUTM has approximate same recall values.

![Recall v/s Number of Nodes in OUTM](image)

**Figure 5.12: Recall v/s Number of Nodes in OUTM**

Figure 5.12 shows the effect of node density over recall. As no. of nodes increases, the Recall value also increases. This happens due to increase in the number of right messages in the network as number of malicious nodes in this case is fixed. So, the accuracy of retrieving truly outlier nodes from the retrieved outlier nodes also increases. OUTM shows an improvement in recall values as compared to both approaches.

![Recall v/s Transmission Range in OUTM](image)

**Figure 5.13: Recall v/s Transmission Range in OUTM**

Figure 5.13 shows the effect of transmission range over recall values. As transmission range increases, the accuracy in identifying outlier nodes increases hence shows an
increase in recall values. However, OUTM shows an improvement as compared to the two approaches.

5.5 Outcome

In this chapter, an outlier detection algorithm is proposed. The working of the proposed outlier detection algorithm (OUTM) is defined by means of a layered structure model. The layered structure model is divided into three layers. Each layer defines the respective part of the algorithm. The algorithm is implemented using MATLAB coding and as a result, outlier nodes are obtained. The performance of the proposed algorithm is measured by calculating three criteria of precision, recall and fmeasure. The comparative analysis with existing algorithms of outlier detection is performed by taking observations in different simulation scenarios by varying simulation parameters. The results obtained show the improvement in accuracy level of output.

Trust generation deals with generating trust values of neighboring nodes based on the proposed trust model. Trust tables are exchanged between neighboring nodes and based on the received tables, every node updates its own table and finally a global table of trust is generated which contains trust values of all the nodes in the network out of which a list of outlier nodes is generated which is the output of the algorithm.