CHAPTER 5

PROPOSED CLUSTER BASED CLONE DETECTION ALGORITHM

This chapter presents the novel method to detect the clone using Cluster Based Clone Detection algorithm (CBCD) in the real-time environment.

5.1 CLONE ATTACK

Assume that an adversary node can reprogram itself by eavesdropping and then act as a legitimate node in the network. Launching various kinds of attacks and taking control over the network is called as clone attack. The clone characteristic is that it drops the packets that it receives. Therefore, the network is susceptible to various kinds of active or passive attacks. Once the clone node joins the network, it injects false data or modifies the data in the network (data integrity is lost). This is known as active attack, whereas in passive attack the clone node just eavesdrops the data (confidentiality is lost). Many centralized and localized protocols have been suggested to address clone attack (Wen et al 2011). While centralized protocols have single point of failure leading to high communication cost, localized protocols do not detect replicated nodes that are distributed in different areas of the network. The miss probability is high in the case of localized protocols.

Detection of clone is a major issue in WSN, because it is difficult to identify when a clone is introduced in the location of different clusters. It is more powerful attack than other existing attacks, because if a node is compromised then it leads to various kinds of attacks. It may cause more damage to the network. Figure 5.1 shows the presence of clone node in a clustered based WSN.

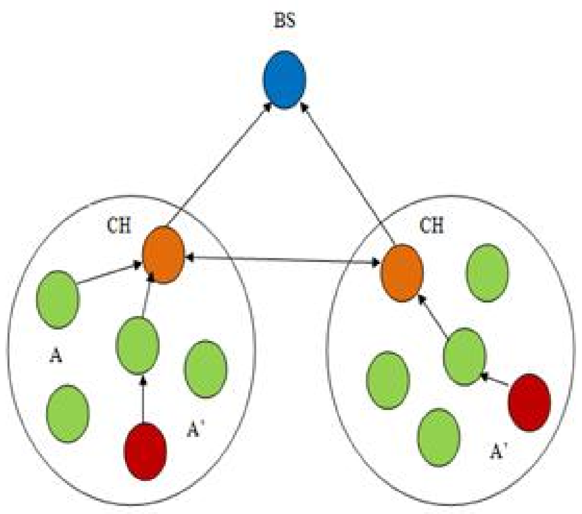


Figure 5.1 Cluster formation in WSN

In Figure 5.1, node A', is a clone of node A. If the clone node is present in the same cluster, then its cluster head identifies the clone node. Suppose if it is in the second cluster the second cluster head has the information about the clone node and when it shares it with the first cluster head the clone node is detected and the details about the presence of clone node is informed to the base station by cluster heads. The base station in turn triggers the revocation procedure of clone node.

The existing clone detection uses group key mechanism (Klaoutatou et al 2011) to detect the clone node. The group key mechanism is more vulnerable to clone attack. This research proposes Cluster Based Clone Detection (CBCD), which uses a shared secret key based scheme to detect clone nodes in the clusters. Every node in the network transmits its location claim to the Cluster Head (CH). A CH acts as a witness node and checks the duplicate node ID and its location claim to detect a clone node. Thus the proposed CBCD algorithm eliminates the clone attacks.

5.2 PROPOSED CLUSTER BASED CLONE DETECTION (CBCD)

In the proposed research work, a Cluster Based Clone Detection (CBCD) approach is used to detect the clone node in the network. In the proposed research, cluster head acts as a witness node. A witness node is a node that identifies a clone that exists in a network. After clusters are formed, each node exchanges its id with its neighbouring nodes in the network and forwards the location claim. Each node maintains a routing table about the neighboring nodes location and id. The architecture of cluster based clone detection is as follows:

- Cluster Formation by Link Estimation Technique
- Route Update and Cost Estimation
- Clone Detection and revocation.

The cluster formation and route updation process are already discussed in chapter 4 under section 4.1 and 4.2. Hence this section discusses about the clone detection and revocation.

5.2.1 Clone Detection and Revocation Process

In this proposed CBCD, a cluster head acts as a witness node that identifies a clone that exists in a network. Witness node receives the claim from its neighboring nodes. If the claim is received for the first time, it is stored in the memory. Otherwise, the received claim is compared with the claim which is stored in the memory; if the values are same, then the claim is accepted. If the values are different, cloned node is identified. This algorithm is executed iteratively at regular intervals by the cluster head so that any clone in the network can easily be detected and revoked thus preventing further attacks. The revocation procedure is done by the base station such that no other legitimate nodes forward their packets to the cloned node and make the clone node idle in the network. The coherence identification process in cluster head is given in Figure 5.2.

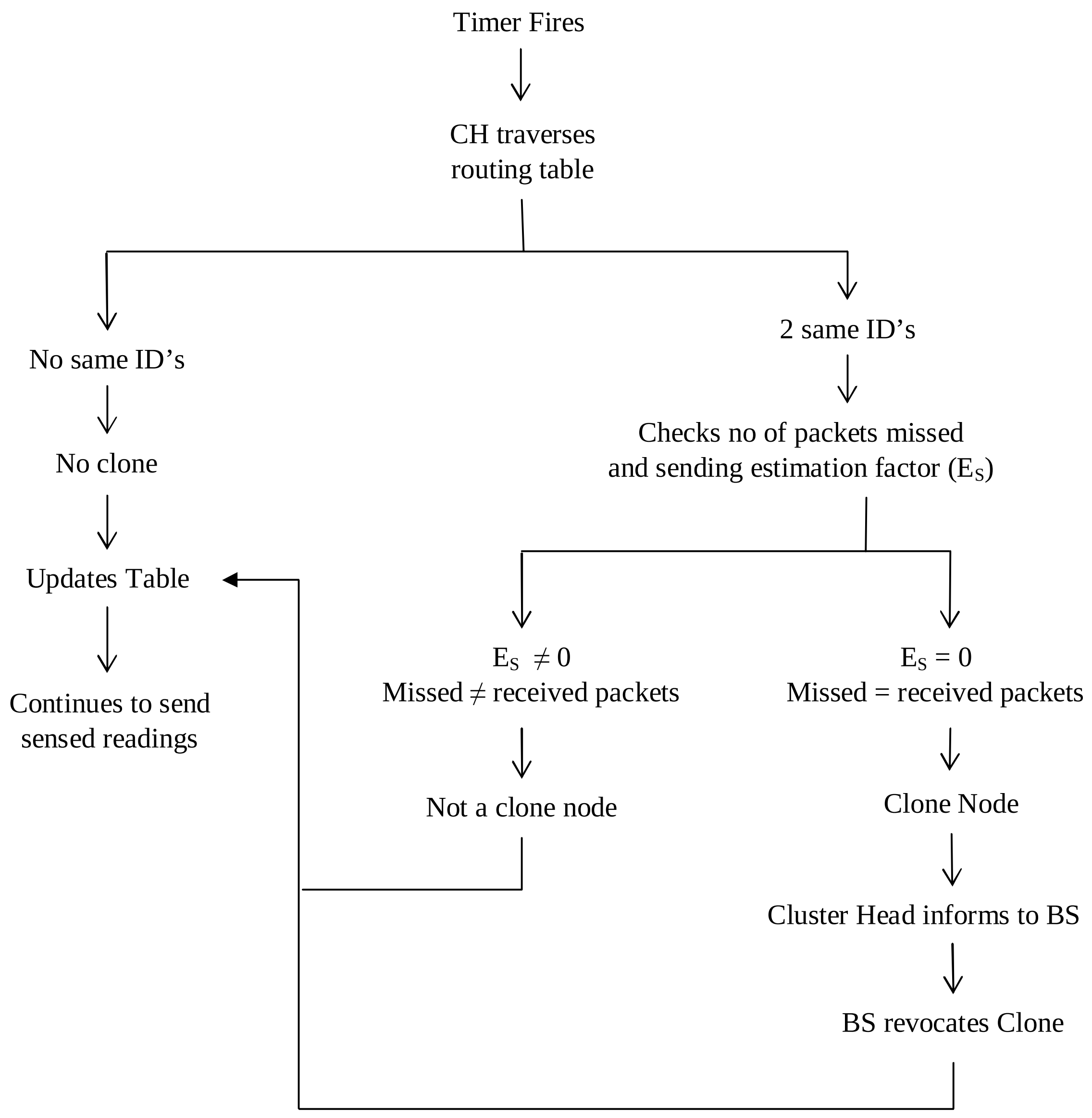

Timer Fires
CH traverses routing table
No same ID's
2 same ID's
No clone
Checks no of packets missed and sending estimation factor (E_S)
Updates Table
$E_S \neq 0$ Missed ≠ received packets
$E_S = 0$ Missed = received packets
Continues to send sensed readings
Not a clone node
Clone Node
Cluster Head informs to BS
BS revocates Clone

Figure 5.2 The coherence identification process

Assume that a clone node in the network drops the packets it receives. Therefore, the sending estimation factor (E_S) becomes zero for the corresponding clone node and the number of packets missed increases in the routing table of the cloned node. If the cluster head encounters two nodes with same id with two location claims then the timer fires and it traverses the routing table to check the number of missed packets and E_s for the two nodes with same id. If the number of missed packets is approximately equal to the number of received packets and $E_S > 128$ then the clone node is identified and it informs to the base station. The base station in turn invokes the revocation procedure and the routing table in cluster head is modified to avoid the clone node during the next route update process.

Algorithm 5.1 The proposed Cluster Based Clone Detection (CBCD) algorithm

```
Node_a :
    $Node_a(Transmit) \rightarrow CH$
            $M := IsClaim(ID_a \| loc_a \| E_{Keya}(H(ID_a \| loc_a)));$
NN_a :
  while ReceiveMessage(M)
  if (M == IsClaim) then
    $WitnessesLocationSet_x \leftarrow CH;$
    {
            $\forall\, loc_{dst}\ \varepsilon\ WitnessesLocationSet_x$
            $NN_a \rightarrow loc_{dst}\ (ID_x \| loc_{dst} \| FwdClaim(ID_x \| loc_x \| SignedClaim_x));$
     }
Cluster Head:
   if IsClaim(M) then
            $Key_x \leftarrow KeysearchLUT(ID_x);$
            if $H(ID_x \| loc_x) \mathrel{!=} D_{Keyx}\ (SignedClaim_x)$
               ExceptionHandler(Error);
            Else
               Authenticated Node;    $Z \leftarrow 0;$
  if $IsNotPresent(MEM,ID_x)$ then
```

```
        AddNewNode(MEM) := Nodex($ID_x$ || $loc_x$ || $SignedClaim_x$);
  else
    LocCheck($loc_x$ || $SignedClaim_x$) ← LookUpLocation(MEM, $ID_x$);
if IsNotCoherent($loc_x$, loc_x)  //coherence identification process shown in Figure 5.2
    Z← 1; FloodNetwork($ID_x$ || $loc_x$ || loc_x || $SignedClaim_x$ || SignedClaim_x);
if  Z ← 0
        accept message from $Node_a$
else
        reject message from $Node_a$
```

CH	Cluster head
D_{Keyx}	Decryption process using shared secret key of $Node_x$
E_{Keya}	Encryption process using shared secret key between $Node_a$ and CH
ID_x	New $Node_x$ ID
Key_a	Secret shared key of $Node_a$
Key_x	Shared secret key of $Node_x$
loc_x	Location of the $Node_x$ is already present in the cluster head
loc_a	Location of $Node_a$
loc_x	Location of new $Node_x$
M	Message
MEM	Memory
NN_a	Neighbors of $Node_a$
SignedClaim_x	Authentication of the $Node_x$ is already present in the cluster head
$SignedClaim_x$	Authentication made by the new $Node_x$

Location claim is the concatenation of node id and its x and y co-ordinates. Before transmitting the claim message, the claim will be encrypted using shared secret key. Encrypted claim is called signed claim. The signed claim is concatenated with id, location (x, y) which is encrypted with the shared secret key and then it is transmitted. If a cluster head is a one-hop neighbor then the node can transmit its location claim directly. If not, the neighbor nodes of that node forward the location claim to the cluster head. The cluster head that receives the claim forwards to all the other cluster heads for claim verification to identify the presence of a clone node. If the node id is present in two different locations, then coherence identification is executed to identify the clone node as shown in figure 5.2. If the node id is not coherence then the node is included as a member of the cluster head else the coherence is informed to CH to validate the trustiness of the node. Once if the behavior of the node is identified as abnormal based on the sending and receiving estimation factor then it will be informed to the BS to call the revocation process to isolate the clone node from the network.

5.3 RESULT AND DISCUSSION OF PROPOSED CBCD

In the proposed research work 50 nodes are randomly deployed. Node id 5 is repeated twice in two different locations which is highlighted in figure 5.3. The clone node is identified using CBCD algorithm. When the clone detection algorithm runs, if the cluster head encounters two nodes with same id in the routing table then it compares the E_R and E_S of both the nodes.

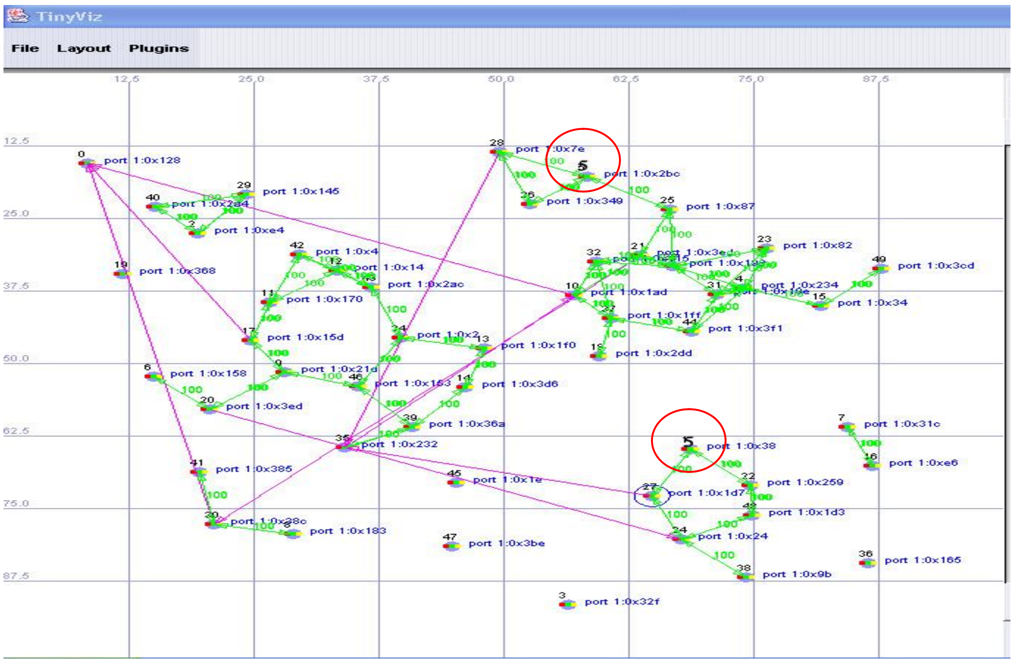
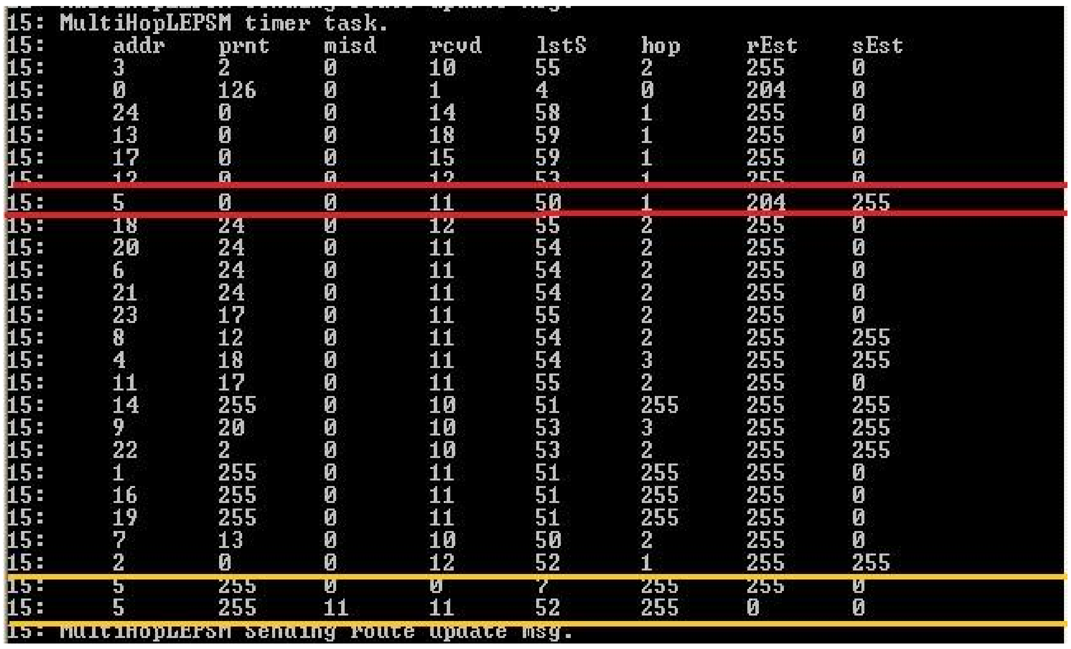


Figure 5.3 Simulation environment with clone node

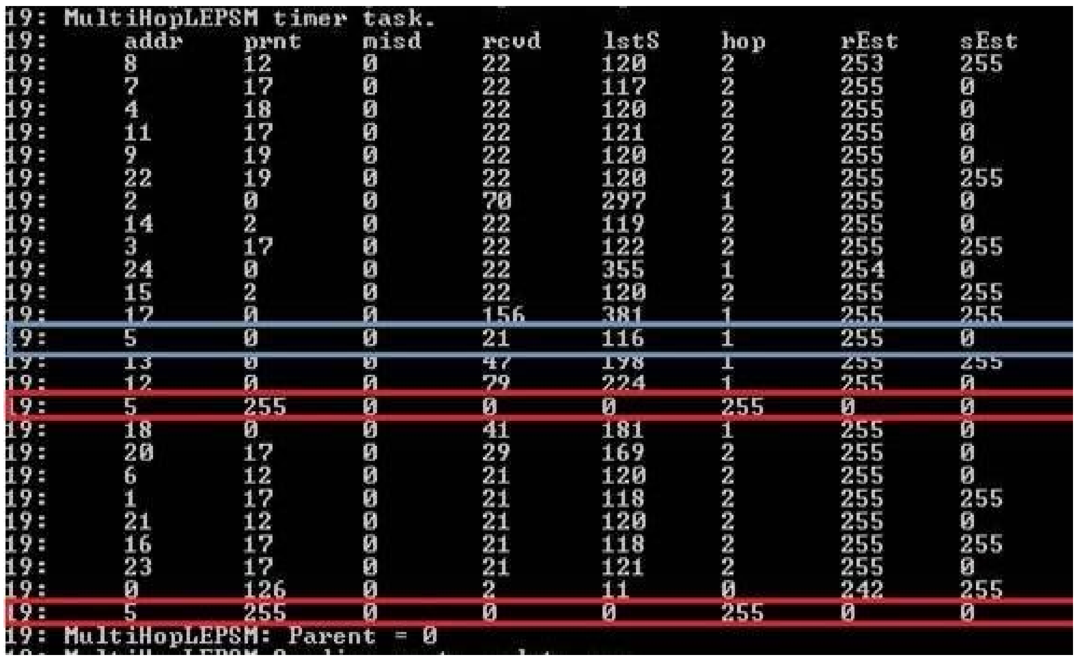
The legitimate node can be distinguished from the clone node using the routing table, which is shown in Figure 5.4. The routing table format is already given in Table 4.2. Here the 5th node is the clone node which receives packets from other nodes but never forwards it to its neighbors. The sending estimation factor (E_S) (shown in column sEst) of the legitimate node 5 is 255, whereas the best sending estimation factor E_S is 0 that indicates the presence of clone node and no packets have been forwarded by it even though its receiving estimation factor (E_R) (shown in column rEst) value is 255. The node with E_R and E_S as 255 and 0 respectively is identified as the clone.



```
15: MultiHopLEPSM timer task.
15:     addr    prnt    misd    rcvd    lstS    hop     rEst    sEst
15:     3       2       0       10      55      2       255     0
15:     0       126     0       1       4       0       204     0
15:     24      0       0       14      58      1       255     0
15:     13      0       0       18      59      1       255     0
15:     17      0       0       15      59      1       255     0
15:     12      0       0       12      53      1       255     0
15:     5       0       0       11      50      1       204     255
15:     18      24      0       12      55      2       255     0
15:     20      24      0       11      54      2       255     0
15:     6       24      0       11      54      2       255     0
15:     21      24      0       11      54      2       255     0
15:     23      17      0       11      55      2       255     0
15:     8       12      0       11      54      2       255     255
15:     4       18      0       11      54      3       255     255
15:     11      17      0       11      55      2       255     0
15:     14      255     0       10      51      255     255     255
15:     9       20      0       10      53      3       255     255
15:     22      2       0       10      53      2       255     255
15:     1       255     0       11      51      255     255     0
15:     16      255     0       11      51      255     255     0
15:     19      255     0       11      51      255     255     0
15:     7       13      0       10      50      2       255     0
15:     2       0       0       12      52      1       255     255
15:     5       255     0       0       7       255     255     0
15:     5       255     11      11      52      255     0       0
15: MultiHopLEPSM Sending route update msg.
```

Figure 5.4 Entries in the routing table of node 15

The revocation procedure is done by the base station by announcing the nodes about the clone nodes id and its location. Therefore, when the route update messages traverses to the base station and route update happens in the routing table the cloned node never receives any packets from the neighboring node and remains idle in the network without participating in any of the communication. Therefore, the E_R and E_S values of the cloned node in the routing table are 0 and 0 respectively. This shows that clone node never receives nor sends any packet as shown in Figure 5.5. The energy overhead for the proposed method is compared with the distributed approach named Random, Efficient and Distributed (RED) algorithm (Conti et al 2011) and is given in the Table 5.1.



```
19: MultiHopLEPSM timer task.
19:      addr     prnt     misd     rcvd     lstS     hop      rEst     sEst
19:      8        12       0        22       120      2        253      255
19:      7        17       0        22       117      2        255      0
19:      4        18       0        22       120      2        255      0
19:      11       17       0        22       121      2        255      0
19:      9        19       0        22       120      2        255      0
19:      22       19       0        22       120      2        255      255
19:      2        0        0        70       297      1        255      0
19:      14       2        0        22       119      2        255      0
19:      3        17       0        22       122      2        255      255
19:      24       0        0        22       355      1        254      0
19:      15       2        0        22       120      2        255      255
19:      17       0        0        156      381      1        255      255
19:      5        0        0        21       116      1        255      0
19:      13       0        0        47       198      1        255      255
19:      12       0        0        79       224      1        255      0
19:      5        255      0        0        0        255      0        0
19:      18       0        0        41       181      1        255      0
19:      20       17       0        29       169      2        255      0
19:      6        12       0        21       120      2        255      0
19:      1        17       0        21       118      2        255      255
19:      21       12       0        21       120      2        255      0
19:      16       17       0        21       118      2        255      255
19:      23       17       0        21       121      2        255      0
19:      0        126      0        2        11       0        242      255
19:      5        255      0        0        0        255      0        0
19: MultiHopLEPSM: Parent = 0
```

Figure 5.5 Revocation process in clone node 5

Table 5.1 Energy overhead

PARAMETERS	**DISTRIBUTED RED**	**PROPOSED CBCD METHOD**
No. of nodes	50	50
Cluster Heads	Nil	10
Number of Witness nodes	50	10
Energy Overhead	O(n)	O(log n/m)
Detection probability	O(n-1)	O(m)

The number of nodes in the network is denoted by n and the number of cluster head is denoted by *m*. It can be seen that in cluster based clone detection method the overhead is only on the cluster heads and it is O(log n/m) whereas in distributed approach the overhead is on all the nodes in the network and it is O(n-1) which reduces the network lifetime.

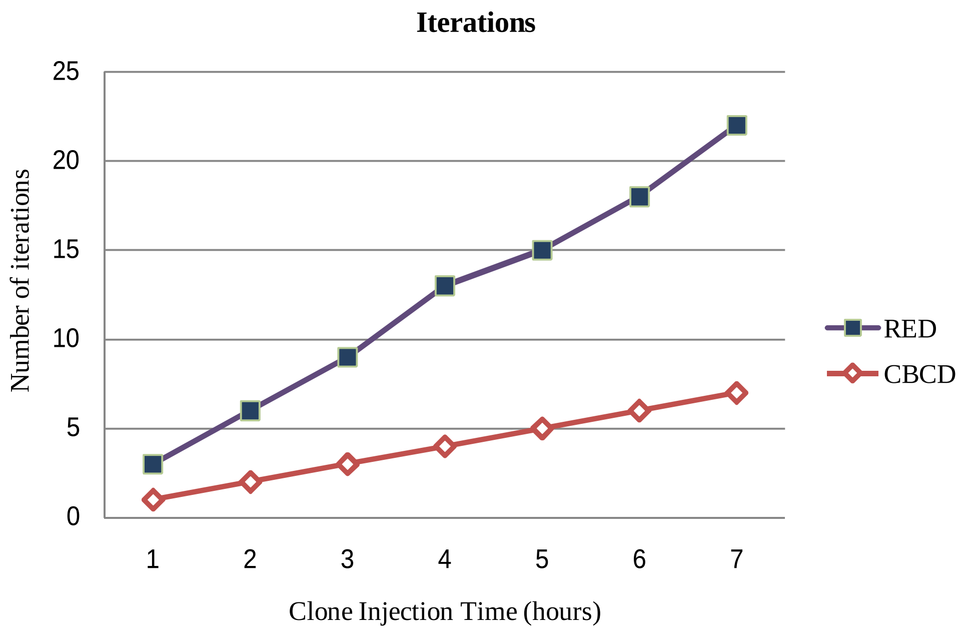


Figure 5.6 Performance analysis of the proposed CBCD algorithm with existing RED algorithm for clone detection

Figure 5.6 shows the performance analysis and comparison of the proposed CBCD based clone detection algorithm and the existing RED based clone detection algorithm in terms of the required number of iterations and time interval. In Figure 5.6, x-axis represents the time interval at which a clone is introduced and y-axis represents the number of iterations to detect the clone node. The RED algorithm requires more iteration and clone detection is possible at the time of forwarding location claim. CBCD algorithm requires minimum time to detect a clone node.

A fixed number of witness nodes are considered in the RED algorithm. However, in CBCD algorithm, the number of witness nodes depends on the network size. Clusters are formed based on the number of nodes in the network. A cluster head acts as a witness node in the network. RED algorithm considers fixed number of witness node as 5 for various size of network. However, in the proposed CBCD the number of witness node is varied based on the size of the network. For example 2 witness nodes are used for a network of size 14, whereas 4 witness nodes are used for 30 number of nodes in the network.

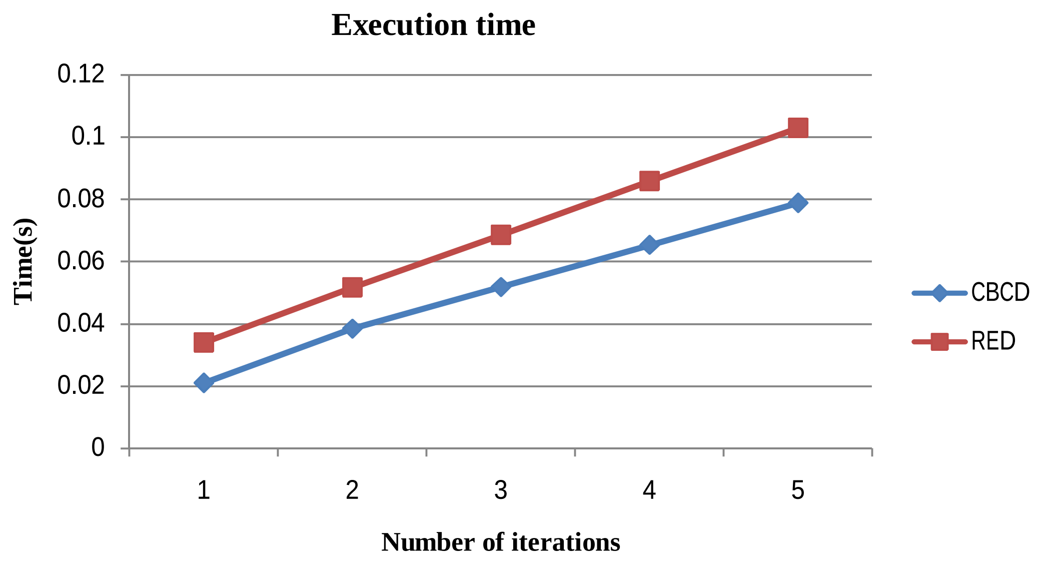


Figure 5.7 Performance analysis of the proposed CBCD algorithm with existing RED algorithm for clone detection

The execution time of the proposed CBCD algorithm and the existing RED algorithm is measured and compared. Figure 5.7 shows the comparison of the proposed CBCD algorithm with the existing RED algorithm for clone detection in terms of the number of iterations and time. In the proposed CBCD the number of witness node depends on the number of cluster heads. The proposed CBCD algorithm consumes lesser time than the existing RED algorithm. A cluster head in the network acts as a witness node for clone detection and reduces the overhead of other nodes.

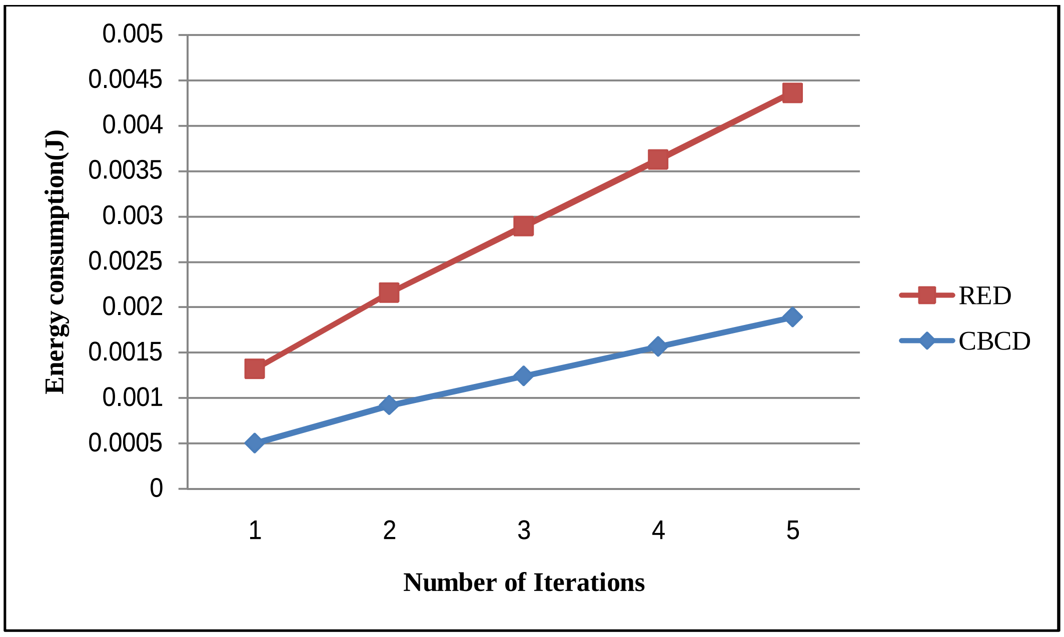


Figure 5.8 Comparison of the energy consumption for the proposed CBCD algorithm and RED algorithm

The energy consumption of the proposed CBCD algorithm and the other existing RED algorithm is measured and compared. Figure 5.8 shows the comparison results. X-axis represents the number of iterations and Y-axis represents the consumed energy in Joules. CBCD algorithm consumes lesser amount of energy than distributed RED algorithm. This is achieved by reducing the overhead by avoiding the transmit and receive of location claims at cluster head.

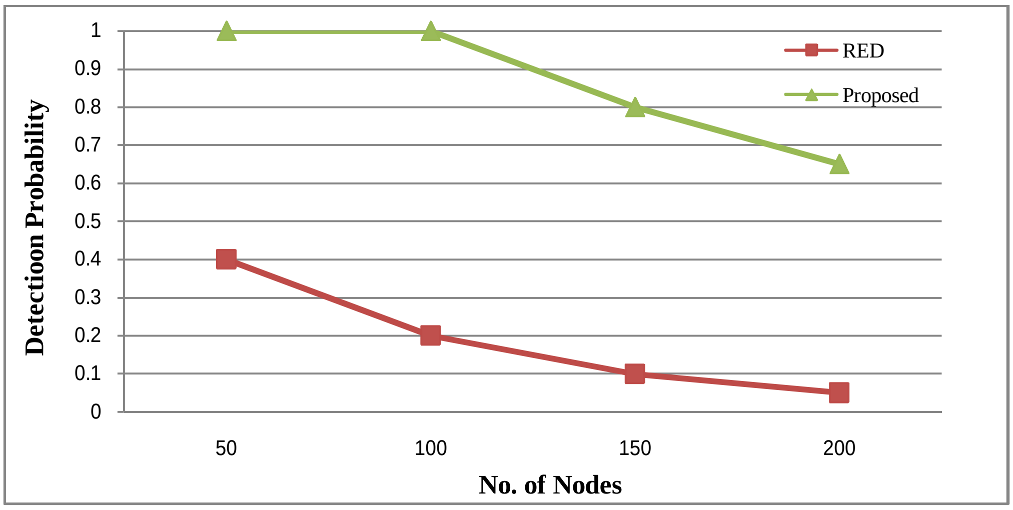


Figure 5.9 Detection probability

Figure 5.9 gives the comparison of the estimated probability of clone detection process in the cluster using the RED based clone detection algorithm suggested by Conti et al 2011 and the proposed CBCD algorithm. It is observed that the performance of the proposed CBCD algorithm provides 100 % results compared to other clone detection algorithm. The energy overhead of the proposed CBCD algorithm is O (log (n/m)) and time complexity of proposed CBCD clone detection is O (m).

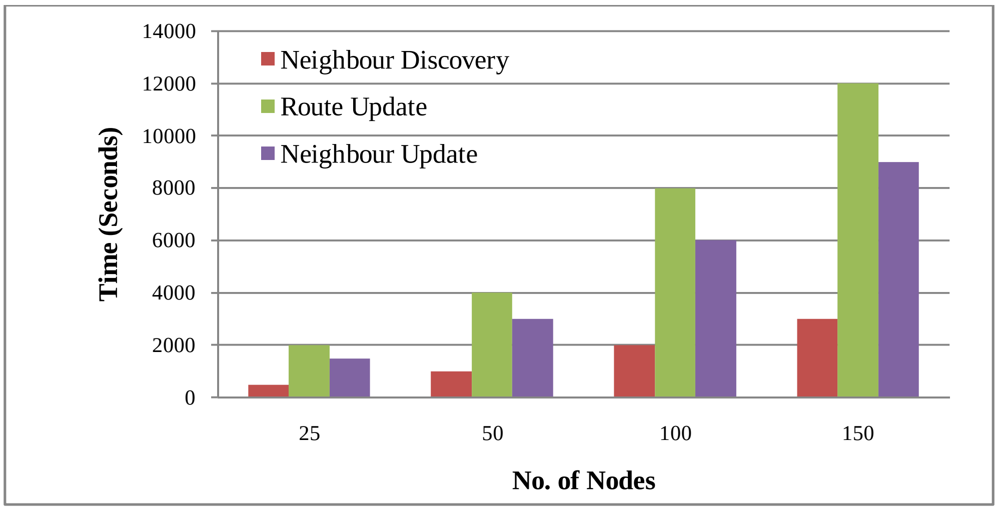


Figure 5.10 Time taken for route update

Figure 5.10 shows the overhead involved in the proposed CBCD clone detection with varying number of nodes. The execution time taken for the proposed CBCD algorithm to detect the clone and update the routing table is proportional to the number of nodes deployed. From the experimental analysis, it is observed that the proposed CBCD clone detection algorithm is suitable for real time implementation.

5.4 SUMMARY

In this chapter, CBCD algorithm is proposed and discussed to identify the clone nodes and the analysis is carried out for cluster and distributed environment. The implementation results are presented and compared with existing RED.