Chapter 6

Design and Development of a Clustered Network

6.1 Introduction

In ad hoc networks, nodes are distributed randomly and they are identified by their unique IDs. In such a network, the execution of a leader election algorithm would result in the identification of only one node. If that node is the only node monitoring all other nodes, then the system no longer remains distributed but becomes centralized. Such a process will result in a single cluster and would require tremendous amount of transmission and computation power. Hence, we have to use a modified version of the leader election scheme. Instead of choosing only one leader, multiple nodes can be elected to perform the duties of the leader. Each of the leaders would be a leader with respect to a subset of the nodes— that is, it would have dominance over a set of nodes. Each leader would know who belongs to the set. This gives rise to the concept of clustering.

A cluster is similar to a ”group” in distributed systems. The nodes within the cluster are considered the member of a cluster head, a node has to lie within the transmission range of that cluster head. However, it is also possible that even if a node is within the transmission range of a cluster head, it may not belong
Development of a Clustered Network

to that cluster head. On the other hand, if a node falls within the transmission range of more than one cluster head, it is called a gateway node. A distributed gateway is a pair of neighboring nodes from different clusters physically located nearest to each other even though their clusters do not overlap.

Most cluster head election algorithms require that each node is a member of one cluster head and no two cluster heads are adjacent to each other. Thus, two cluster heads cannot be one-hop neighbors. Therefore, ad hoc networks make use of multiple clusters and require multiple cluster heads. Clusters may change dynamically, reflecting the mobility of the underlying network. The focus of the existing literature in this area has mostly been on partitioning the network into cluster [65], without taking into consideration the efficient functioning s of all the system components. Due to this dynamic nature of the mobile nodes, their association and dissociation to and from clusters perturb the stability of the network and thus reconfiguration of the cluster heads is unavoidable. This is an important issue since frequent cluster head changes adversely affect the performance of other protocols such as scheduling, routing and resource allocation that rely on it. The lack of rigorous methodologies applicable to the design and analysis of peer-to-peer mobile networks has motivated in-depth research in this area. There have been solutions for efficient ways of interconnecting the nodes such that the latency of the system is minimized while throughput is maximized [66]. Most of these approaches [67], [63], [66] for finding the cluster heads do not produce an optimal solution with respect to battery usage, load balancing and MAC functionality. Since choosing cluster heads optimally is an NP-hard problem [67], existing solutions to this problem are based on heuristic (most greedy) approaches and none of the attempts to retain the stability of the network topology [67].

We believe that a good clustering scheme should preserve the graph structure as much as possible when nodes are moving and/or the topology is slowly changing. Otherwise, re-computation of cluster heads and frequent informa-
tion exchange among the participating nodes will result in high computation
and communication overhead. Thus, optimal selection of cluster heads and
partitioning of the nodes into clusters are essential aspects of mobile ad hoc
networks.

6.2 Efficient non-overlapping clusters

To achieve efficient non-overlapping clusters based on unique Hop ID, the fol-
lowing features are to be addressed and solved. The algorithms are developed
and presented here to construct strictly non overlapping clusters.

1. Generate unique ID for each and every node of a network (LMUA).

2. Generate unique ID which satisfies the cluster dimension property to form
   non-overlapping clusters (LMUAC).

3. Identify those nodes which partition the network to form non-overlapping
   clusters (LMUAC).

4. Classify the nodes in a cluster (LMUAC).

5. Identify the gateway nodes in a network (LMUAC).

6. Generate unique code for CHs at 1 level hierarchy (LMUAC).

7. Switching between CHs at level 1 (w.r.t Source and Destination Hop
   ID) (LMUAC).

The below addressed algorithm explores the following features

• It operates efficiently in generating unique clustering Hop ID during off
  line condition ($N – Tuple$) by (LMUAC).

• It provides full name based addressing, thus accommodating a percentage
  of nodes.
• It provides automatic address assignment, thus easing network administration.

• It accommodates administrative boundaries, providing control of routing paths, protection and autonomy.

• It helps in Formation of a non-overlapping clusters.

• It helps in identifying the type of nodes.

• Communication through cluster heads is adopted.

The purpose of an efficient Hop ID for clustering is to optimize routing in very large networks, the control is distributed in the networks, and the clustering Hop ID accomplishes this.

### 6.3 Basic initialization

#### 6.3.1 N-Tuple

**Unique ID** is an $N-Tuple v(d_1, d_2, \ldots, d_n)$ associated with every node of the network where $d_i$ represents the distance of the node from the $i^{th}$ cluster head $CH_i$ for $i = 1, 2, \ldots, n$.

*Note 6.1.* The $i^{th}$ position value of length of the $N-Tuple$, say $N$, the $N-Tuple$ associated with the cluster head $CH_i$ is zero. For $i = 1, 2, \ldots N$ defines the number of cluster heads in the cluster basis. The length of the Tuple, say $N$, is the cluster dimension of $G$.

#### 6.3.2 Cluster

Clustering a set $S$ is dividing the set into smaller subset $s_1, s_2, \ldots s_N$ such that $s_1 \cup s_2 \cup, \ldots s_N = S$. Each set $s_i$ is called a clusters.
In our case, the $i^{th}$ cluster is the collection of all the vertices which are at minimum distance from the $i^{th}$ cluster head $1 \leq i \leq n$. Equivalently it is the collection of these $N-Tuples$ which have the least entry in the $i^{th}$ position $1 \leq i \leq n$.

**Note 6.2.** The clusters obtained by the above procedures are non-overlapping and non empty. The classification of the nodes inside a cluster using the $N-Tuple$ associated with the nodes as follows:

### 6.3.3 Classification of Nodes

1. **Cluster Head:** It is that node in the cluster which contains a 0 in its $N-Tuple$

2. **Member Node:** Every node which has the least entry in the $i^{th}$ position of its $N-Tuple$ is a member node of $i^{th}$ cluster, $i = 1, 2, \ldots, n$

3. **Gateway Node:** A gateway node in the $i^{th}$ cluster, $i = 1, 2, \ldots, n$ is a member node in that cluster which has an adjacent node in the $j^{th}$ cluster, $i \neq j$, $j = 1, 2, \ldots n$

Let $n$ be the number of vertices in $V(G)$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$ be the vertex set of $G$. Let $\{v_1, v_2, \ldots, v_N\}$ be the landmarks in the network. For any vertex $v$, $(Co-ord)_j$ is the $j^{th}$ Co-ord in the $N-Tuple$ for $v$ and it is equal to $d(v, v_j)$, $1 \leq j \leq N$. For every vertex $v$, $N-Tuple(v)$ is the $N-Tuple$ of co-ordinates associated with $v$ which is treated as address of $v$ or Unique ID for $v$. Let $\beta(G)$ be the metric dimension of $G$, $m$ is the metric basis of $G$, where $N$ is an integer which is greater than or equal to metric dimension (number of landmarks), $N \geq |m|$. Where $Pos$ is defined as the value of the integer $t$ for which the $t^{th}$ Co-ord is minimum in Co-ord of $v$ and $C_{min}$ is defined as, if $Pos$ value is $k$, the value of the position $v_i$ is the value of $k^{th}$ Co-ord in $N-Tuple(v)$. 


6.4 Input Distance Matrix

The input of the network is taken in terms of the distance between two nodes rather than the adjacency or incidence. The purpose of representation of a network in terms of its distance matrix can be easily adopted for any type of network because the distance can be considered in various ways depending on the purpose in hand. For example the distance between two stations can be considered as air distance or for the land lines it may be length of the wire connection between two junctions to identify the fault in an exact location. The input can be easily modified for a weighted graph by considering the sum of weights on the edges in the shortest path as the distance between the corresponding nodes in the network. Throughout the algorithm we assume the network is a connected and represented by a square matrix $D = dist[i][j]$ of order $n$, where $n$ is the number of nodes in the network and $dist[i][j]$ denotes the distance between the two nodes $i$ and $j$. For any network the matrix $D$ is a symmetric matrix with zero diagonal entry if and only if the network is undirected. The Algorithm 6.4.1 is developed which is shown below.

6.4.1 Algorithm for Input Distance Matrix

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>Initialize dist[n][n]</td>
</tr>
<tr>
<td>2:</td>
<td>Step 0: Reserve the memory for storing the distance:</td>
</tr>
<tr>
<td>3:</td>
<td>n × n matrix is reserved</td>
</tr>
<tr>
<td>4:</td>
<td>Step 1: Read the distance matrix:</td>
</tr>
<tr>
<td>5:</td>
<td>For i=2 to n .................Column</td>
</tr>
<tr>
<td>6:</td>
<td>For j=1 to i-1 .................Row</td>
</tr>
<tr>
<td>7:</td>
<td>dist[i][j] ← Value</td>
</tr>
<tr>
<td>8:</td>
<td>Exit</td>
</tr>
<tr>
<td>9:</td>
<td>Step 2: Output</td>
</tr>
<tr>
<td>10:</td>
<td>dist[i][j] is the distance between $i^{th}$ and $j^{th}$ vertex</td>
</tr>
<tr>
<td>11:</td>
<td>$1 \leq i \leq n$, $1 \leq j \leq i$</td>
</tr>
</tbody>
</table>
6.5 Generating Unique ID for vertices in a graph $G$ (LMUA)

Throughout this chapter, the graph is written as $G = (V, E)$ or simply $G$, to denote a graph on a finite non empty set $V$ of vertices and $E$ of edges. All the graphs considered in this paper are simple, finite, undirected and connected. For any two vertices $u$ and $v$, the distance between $u$ and $v$ denoted by $d(u, v)$ is the length of the shortest path between them. The vertices in a graph $G$ are to be identified uniquely in a given graph $G$. The purpose is to generate unique Hop ID ($N$ − Tuple) which is a multi coordinate system with respect to the pre selected landmarks. The co-ordinates associated with each node is based on the distance from the node to the randomly selected landmarks, the randomly selected landmarks must be just enough to generate a unique Hop ID ($N$ − Tuple) of co-ordinates. The coordinates of $v_i$ vertices in a network with respect to $N$ Landmarks are represented by $(N$ − $Tuplev_i) = ((Co−ordv_i)_1,(Co−ordv_i)_2,\ldots,(Co−ordv_i)_N)$, For $1 \leq j \leq N$, $1 \leq i \leq n$, where $(Co−ordv_i)_N = d(v_i, u_j)$, where $u_j \leftarrow N^{th}$Landmark.

A co-ordinate system on $G$ is defined as follows. Select a set $N$ of nodes as the landmarks. For each landmark, the co-ordinate of a node $v_i \in V$ in $G$ having the elements equal to the cardinality of the set $N$ and $j^{th}$ element of co-ordinate of $v_i$ is equal to the length of shortest path from the $j^{th}$ landmark to the vertex $v_i$ in $G$. Here onwards Hop ID and $(N$ − $Tuple)$ are interchangeably used for our convenience.

The aim of this Algorithm 6.5.1 is to find the minimum number of Land Marks (by randomly selected) required to generate Unique Hop ID Addressing (LMUA) for each and every node, which is shown below. The input is a distance matrix and initially let us assume that it has selected $j$ landmarks and is still not able to generate the unique Hop ID. The Algorithm 6.5.1 is extended to generate unique Hop ID as follows. In step 0: The initialization has been
Development of a Clustered Network

carried out. In step 1: Selection of new landmark \( u_{j+1} \) is carried out randomly from the remaining set of vertices \( \hat{V} \), the number of landmarks is increased, \( N++ \), the size of the metric basis element is also increased by adding the vertex \( u_{j+1} \) to its set. In step 2: The generated unique Hop ID with respect to the landmarks which are unique is verified. In step 3, the Algorithm 6.5.1 explores all the metric basis elements, minimum number of landmarks required which is \( \geq \beta(G) \) and unique Hop ID generated for each and every vertex in a graph \( G \) which is essential for our purpose.

6.5.1 Algorithm for Metric Dimension(LMUA)

. 1: Input distance matrix
. 2: \((Co - ordv_i)_j = d(v_i, u_j)\)
. 3: \(j=1,2,3,\ldots N\)
. 4: **Step 0:** .Initialize
. 5: .If the graph is not a path then \( \beta(G) > 1 \)
. 6: .Suppose that \( Co - ord_i = i^{th} \) Co-ord of a vertex, \( 1 \leq i \leq N \)
. 7: \( N - TupleV_i = ((Co - ordv_i)_1, (Co - ordv_i)_2, \ldots, (Co - ordv_i)_j) \)
. 8: is already exist, \( 1 \leq j \leq N, 1 \leq i \leq n \)
. 9: \( N \rightarrow \geq \) Metric Dimension
. 10: \( n \rightarrow \) Number of vertices in a graph \( G \)
. 11: \( m \rightarrow \) Metric basis elements set= \{\( u_1, u_2, \ldots, u_j \)\}
. 12: \( V_i \rightarrow \) Total number of Vertices in a graph \( G \)
. 13: \( V_i = \{v_1, v_2, \ldots, v_n\}, 1 \leq i \leq n \)
. 14: \( V' = V - m, \) remaining number of vertices for selection of Metric basis
. 15: **Step 1:** Selection of Metric Basis Element to generate unique ID.
. 16: Select a vertex \( u_{j+1} \) at random from \( \hat{V} \), \( N = j + 1, m\{\} \leftarrow m\{\} \cup u_{j+1} \)
. 17: \( m = \{u_1, u_2, \ldots, u_j, u_{j+1}\} \)
. 18: \([C{(o - ordu_{j+1})} = 0]\)
. 19: Generate N Tuple \( v_i, N = j+1, 1 \leq i \leq n \ldots Code for rest of vertices
. 20: **Step 2:** Identify Unique ID.
. 21: For \( 1 \leq i \leq n \)
 22: \((Co - ordv_i)_{j+1} = d(v_i, u_{j+1})\)
. 23: For \( 1 \leq i \leq n - 1 \)
 24: For \( i + 1 \leq i < n \)
 25: If \((NTuple \ v_i) == (NTuple \ u'_i)), i < i\)
 26: \{N++, Go to step 1\}
. 27: Else Exit
. 28: **Step 3:** Output
. 29: \( m = \{u_1, u_2, \ldots, u_j, u_{j+1}\} \ldots \ldots \ldots Metric Basis Elements.\)
. 30: \( N = \{int\} \geq \beta(G) \ldots \ldots \ldots Metric Dimension.\)
. 31: N Tuple \( v_i, 1 \leq i \leq n \ldots \ldots \ldots Unique ID for all the vertices of \( V.\)
The following example demonstrates how to generate unique Hop ID for each and every network for a given graph of $G$. Let $L_1, L_2$ and $L_3$ be the landmarks selected in a graph $G$ of 29 vertices to generate unique Hop ID, let $N = 3$, $m = \{L_1, L_2, L_3\}$ the first Co-ord associated to each and every node is the shortest distance from first landmark i.e, $L_1$, the generated unique $(N-Tuple_{v_i})$ is not unique we know that the graph is not a path. So the second landmark $L_2$ is selected randomly from the remaining set of vertices of $G$ and the second Co-ord associated to each and every node is the shortest distance from the second landmark i.e, $L_2$ the generated unique $(N-Tuple_{v_i})$ is also not unique, so the third landmark $L_3$ is randomly selected from the remaining set of vertices of $G$ and the length of the Co-ord is three and the third Co-ord is the distance from the third landmark which satisfies the concept of metric dimension, where $N \geq \beta(G)$, which is shown in Figure [6.1] Which gives the upper bound of the metric dimension. The output of the LMUA Algorithm is shown in the Table

![Figure 6.1: Unique Hop ID by LMUA](image)

100
Table 6.1: Output of the LMUA Algorithm

<table>
<thead>
<tr>
<th>Characteristics of the LMUA Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept</td>
</tr>
<tr>
<td>$n$</td>
</tr>
<tr>
<td>$N$</td>
</tr>
<tr>
<td>$m$</td>
</tr>
<tr>
<td>$N$</td>
</tr>
<tr>
<td>$(N - \text{Tuple } u)$</td>
</tr>
<tr>
<td>Clusters</td>
</tr>
<tr>
<td>Disadvantages</td>
</tr>
<tr>
<td>Applications</td>
</tr>
<tr>
<td>Complexity</td>
</tr>
<tr>
<td>Complexity $\beta(G) = 2$</td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>$</td>
</tr>
</tbody>
</table>

### 6.6 Clustering the graph using Hop ID

A successful approach for dealing with the maintenance of mobile ad hoc networks is by partitioning the network into clusters. In this way the network becomes more manageable. It must be clear though that a clustering technique is not a routing protocol. Clustering is a method which aggregates nodes into groups [68]. These groups are contained in the network and they are known as clusters. A cluster is basically a subset of nodes of the network that satisfies a certain property [69]. Clusters are analogous to cells in a cellular network. However, the cluster organization of an ad hoc network cannot be achieved off line as in fixed networks [70]. Clustering presents several advantages for the
Development of a Clustered Network

medium access layer and the network layer in MANET [39]. The implementation of clustering schemes allows a better performance of the protocols for the Medium Access Control (MAC) layer by improving spatial reuse, throughput, scalability and power consumption. On the other hand, clustering helps improve routing at the network layer by reducing the size of the routing tables and by decreasing transmission overhead due to the update of routing tables after topological changes occur [71], [69]. Clustering helps aggregate topology information since the number of nodes of a cluster is smaller than the number of nodes of the entire network. Therefore, each node only needs to store a fraction of the total network routing information [72].

Figure 6.2: Clustering by unique hop ID (LMUA)

Our proposed routing protocol is based on the formation of clusters, hence efficient cluster formation will be the crux of the routing protocol of this nature. Cluster should be formed in such a way that the resulting graph is cluster connected. Routing from one node to another will consist of routing inside a cluster and routing from cluster to cluster. The $N-Tuple_{v_i}$ generated for each and every vertices in a graph $G$ for the above example is unique, for $1 \leq i \leq n$. 

The vertices of a graph $G$ are to be strongly resolved by one and only one landmark by the definition of the cluster dimension. The strongly resolving set is considered are as follows, the nodes are strongly resolved by the landmark $L_1$ if the value of the first digit in the $N$ – $Tuple_i$ Hop ID is the least. Similarly the nodes are strongly resolved by a landmark $L_2$ if the value of the second digit in the $N$ – $Tuple_i$ $v_i$ is the least. This process is carried out for all the landmarks and the resultant of the above graph is the formation of the overlapping clusters which is a draw back in the clustered network. The nodes which are not strongly resolved by one of the landmarks; therefore the role of that member node becomes difficult (like making decision, control overhead) as to with which cluster it has to communicate and, if it is resolved by more than one landmarks the control is with more than one landmarks so it is a complicated member node. To avoid such type of nodes in a network we should satisfy the cluster dimension property. The above Figure 6.2 demonstrates that the unique ID generated using the concept of the metric dimension is insufficient.

6.7 Definition: Clustering

1. Let $G = (V, E)$ be a graph representing the given network. A clustering is a specific method of dividing the vertex set $V(G)$ according to the convenience of the user. For our purpose, the clusters are non intersecting, non empty subsets of $V(G)$. whose union is equal to $V(G)$.

2. The basic tool to introduce clusters in $V(G)$ is a 'metric' which is the shortest hop count distance between the nodes.

3. To every vertex, a unique clustering ID is generated which satisfies the properties given in definition 4.1.

4. The $J^{th}$ cluster, say $C_j$ contains vertex $v_i$, with ID

$$((Co – ordv_i)_1, (Co – ordv_i)_2, \ldots, (Co – ordv_i)_N),$$
such that \((Co - ordv_j)_j < (Co - ordv_k)_k\) \(\forall k \neq j, k = 1, 2, 3, \ldots, N\).

The concept of metric dimension is extended by adding an additional property that each unique ID is to be strongly resolved by one and only one cluster. To achieve this an additional landmark \(L_4\) is randomly selected to generate unique clustering Hop ID for clustering the network. The following algorithm generate unique clustering Hop ID which satisfies the definition of cluster dimension. This algorithm explores all the cluster basis elements, Cluster Dimension and \(N - Tuplev_i\), for \(1 \leq i \leq n\).

**Theorem 6.3.** Between any two cluster heads, if there is a path of length two then the vertex which is at a distance one from both the cluster heads must be another cluster head.

**Proof.** Let there be a path of length two between the cluster heads say \(v_i \& v_j\). Let \(u\) be adjacent to both \(v_i \& v_j\) on the above path. If \(u\) is not a cluster head, then \(u\) is not strongly resolved by any of the vertices in \(G\) by Definition of 6.3. Hence \(u\) must be a cluster head. \(\square\)

**Theorem 6.4.** Between any two clusters heads if there is a path of length three then the intermediate vertices on the path are the gateway nodes.

**Proof.** Similar to the above proof 6.3. \(\square\)

**Note 6.5.** On any path between the given two cluster heads there is a non-gateway member node. If the length of the path is greater than three.

### 6.7.1 Algorithm for Cluster Dimension (LMUAC)

The aim of this Algorithm which is shown below is to generate Unique ID for each and every node which is dominated by one and only cluster head. To satisfy this condition the above Algorithm 6.5.1 developed is insufficient, which constructs either overlapping/non-overlapping clusters, which makes those nodes
which are dominated by more than one cluster heads. The output of this algorithm is shown below, which has generated an unique ID.

1: Input distance matrix
2: \((Co - ord v_i)_j = d(v_i, u_j)\)
3: \(j=1,2,3, \ldots N\)
4: **Step 0**: Initialize
5: .If the graph is not a path then \(\beta_c(G) > 1\)
6: .Suppose that \(Co - ord_i = \text{ith Co-ord of a vertex, } 1 \leq i \leq N\)
7: \(\text{N-Tuple}\ V_i = ((Co - ord v_i)_1, (Co - ord v_i)_2, \ldots, (Co - ord v_i)_j)\)
8: is already exist, \(1 \leq j \leq N, 1 \leq i \leq n\)
9: \(N \rightarrow \geq\) Cluster Dimension
10: \(n \rightarrow\) Number of vertices in a graph G
11: \(m \rightarrow\) cluster basis elements set= \(\{u_1, u_2, \ldots, u_j\}\)
12: \(V_i \rightarrow\) Total number of Vertices in a graph G
13: \(V_i = \{v_1, v_2, \ldots, v_n\}, 1 \leq i \leq n\)
14: \(V' = V - m,\) remaining number of vertices for selection of cluster basis
15: **Step 1**: Select cluster basis Element to generate unique ID from \(V'.\)
16: Select a vertex \(u_{j+1}\) at random from \(V', N = j + 1\)
17: \([(Co - ord u_{j+1}) = 0]\)
18: \(m\{\} = m\{u_1, u_2, \ldots, u_j\} + u_{j+1}\)
19: \(m = \{u_1, u_2, \ldots, u_j u_{j+1}\}\)
20: Generate N Tuple \(v_i, N=j+1, 1 \leq i \leq n\). Code for rest of vertices
21: **Step 2**: Generate \(C_{min}\) to satisfy Cluster Dimension Definition.
22: For \(1 \leq i \leq n,\) False=0, pos
23: \(C_{min} = (Co - ord N Tuple \\ v_i)_1\)
24: For \(2 \leq j \leq N\)
25: If \((C_{min} = (NTuple \\ v_i)_j)\)
26: \{False=1\}
27: ELSE IF \((C_{min} < (NTuple \\ v_i)_j)\)
28: \{continue\}
29: ELSE \(C_{min} \leftarrow (NTuple \\ v_i)_j\), False=0, pos=j
30: If (False=1), go to step 1
31: ELSE
32: \((Co-ord N Tuple \ v_i)(N + 1) \leftarrow pos\)
33: **Step 3**: Identify Unique ID.
34: For \(1 \leq i \leq n\)
35: \((Co - ord v_i)_{j+1} = d(v_i, u_{j+1})\)
36: For \(1 \leq i \leq n - 1\)
37: For \(i + 1 \leq i \leq n\)
38: If \(((NTuple \ \ v_i) == (NTuple \ \ v_i)_j), i < \hat{i}\)
39: \{N++, Go to step 1\}
40: Else Exit
41: **Step 4**: Output
42: \(m = \{u_1, u_2, \ldots, u_j, u_{j+1}\}\) .......Cluster Basis Elements.
43: \(N = \{\text{int} \} \geq \beta_c(G)\) ..........Cluster Dimension.
44: N Tuple \(v_i, 1 \leq i \leq n\) .......Unique ID for all the vertices of \(V.\)
45: pos .................the position value of \(C_{min}\) in N Tuple \(v_i\).
6.8 Formation of non-overlapping clusters.

The aim is to form non-overlapping clusters for an efficient communication, which leads to reduction in routing Table size, communication overhead and information of only neighbor nodes. The following algorithm is used to carry out strictly non-overlapping clusters with the help of unique clustering Hop ID which is shown in the Figure below which satisfies cluster dimension property. The unique $N - Tuple_v_i$ of the vertices for $1 \leq i \leq n$ is clustered. The first $(Co - ordv_i)$ is least then they are all strongly resolved by the first landmark $L_1$ and available in $C_1$ and the second $(Co - ordv_i)$ is least then it is strongly resolved by the second landmark $L_2$ and available in $C_2$. Similarly the procedure follows for all the landmarks. None of the nodes are strongly resolved by more than one of the landmarks and none of the member nodes belong to more than one cluster. The resultant is, it forms non-overlapping clusters. Each and every
cluster has one and only one landmark and the resultant is \( C_1 \cup C_2 \cup C_3 \cup C_4 = V \).

### 6.8.1 Algorithm for construction of Non-Overlapping Clusters

The aim of this below Algorithm is to construct clusters based on Unique Identity number generated for each and every node by LMUAC Algorithm. Collection of those nodes which are dominated by one of the cluster head, i.e, the position of the least number in the unique Id implies that node to which cluster it belong. The construction of non-overlapping clusters is carried out which is shown in Figure 6.4. The output of the LMUAC is listed in the below Table 6.2.

6.2

1: **Step 0:** Input is \((\text{Coord NTuple } v_i)_{N+1} = \text{pos}\)
2: \((\text{NTuple } v_i)_{N+1} = ((\text{Coord } v_i)_1, (\text{Coord } v_i)_2, \ldots (\text{Coord } v_i)_N, \text{pos})\)
3: \(C_i = \{\}, 1 \leq i \leq N\)
4: **Step 1: Vertex is clustered**
5: For \(1 \leq i \leq N\)........Cluster Dimension
6: For \(1 \leq j \leq n\)........No of vertex in a Graph
7: if\(((\text{Coord Ord } N - \text{Tuple } v_i)_{N+1} == i)\)
8: \(C_i\{\} \leftarrow v_j\)
9: **Step 2:** Output.
10: \(C_1 = \{\text{nonemptyset}\}, C_2 = \{\text{nonemptyset}\}, \ldots, C_N = \{\text{nonemptyset}\}\)
11: \(C_1 \cup C_2 \cup C_3, \ldots \cup C_N = V\).

### 6.9 Classification of nodes in a cluster based on Hop ID

Clustering is a method which aggregates nodes into groups [68]. These groups are contained in the network and they are known as clusters. A cluster is basically a subset of nodes of the network that satisfies a certain property. A cluster is a collection of nodes like Cluster Head and a Member node. These nodes are classified into member node and cluster head. Member nodes are members of a cluster which do not have neighbors belonging to a different cluster. Most clustering approaches for mobile ad hoc networks select a subset
of nodes in order to form a network backbone that supports control functions. A set of the selected nodes are called cluster heads and each node in the network is associated with one. Cluster heads are connected with one another directly or through gateway nodes. The union of gateway nodes and clusterheads form a connected backbone. This connected backbone helps simplify functions such as channel access, bandwidth allocation, routing power control and virtual-circuit support [73]. Clusterheads are analogous to the base station concept in current cellular systems. They act as local coordinators in resolving channel scheduling and performing power control [66]. However, the difference of a clusterhead from a conventional base station resides in the fact that a clusterhead does not have special hardware, it is selected from among the set of stations and it presents a dynamic and mobile behavior [74]. Since clusterheads must perform extra work with respect to ordinary nodes they can easily become a single point of failure within a cluster. For this reason, the clusterhead election process should consider for the clusterhead role, those nodes with a higher degree of relative
Table 6.2: Output of LMUAC Algorithm

<table>
<thead>
<tr>
<th>Characteristics of the LMUAC Algorithm</th>
<th>Strongly Resolving Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept</td>
<td>Strongly Resolving Set</td>
</tr>
<tr>
<td>( n )</td>
<td>29</td>
</tr>
<tr>
<td>( N )</td>
<td>4</td>
</tr>
<tr>
<td>( m )</td>
<td>( {L_1, L_2, L_3, L_4} )</td>
</tr>
<tr>
<td>( \hat{N} )</td>
<td>( \geq \beta_c(G) )</td>
</tr>
<tr>
<td>((N - Tuple \ u))</td>
<td>Unique</td>
</tr>
<tr>
<td>Clusters</td>
<td>Non Overlapping</td>
</tr>
<tr>
<td>Disadvantages</td>
<td>Number of landmarks is more</td>
</tr>
<tr>
<td>Applications</td>
<td>Cluster for communication,</td>
</tr>
<tr>
<td></td>
<td>Hierarchical routing</td>
</tr>
<tr>
<td>Complexity</td>
<td>( NP ) hard problem -General</td>
</tr>
<tr>
<td>Complexity ( \beta_c(G) = 2 )</td>
<td>( \left( \frac{n(n-1)}{2} D^4 \right) )</td>
</tr>
<tr>
<td>(</td>
<td>V</td>
</tr>
<tr>
<td>(</td>
<td>V</td>
</tr>
</tbody>
</table>

stability [68]. The main task of a clusterhead is to calculate the routes for long-distance messages and to forward inter-cluster packets. A packet from any source node is first directed to its clusterhead. If the destination is in different clusters then the switching between the cluster heads is carried out.

### 6.9.1 Algorithm for Classification of nodes in a cluster

1: Cluster Head=\( \phi \rightarrow CH_j\{ \} \), Member node=\( \phi \rightarrow MN_j\{ \} \)
2: **Step 0: Input** \( C_j = \{w_{j1}, w_{j2}, w_{j3}, \ldots w_{jk}\} \)
3: the vertices \( \in C_j \), all these vertices are having
4: \( C_{min}^{inj} \) position of an N Tuple \( 1 \leq j \leq N \).
5: **Step 1:** For \( 1 \leq j \leq N \)
6: For \( 1 \leq t \leq k \)
7: if\( ((C_0 - ordw_{jt})_j == 0) \)
8: \( \{CH_j = \{\} \leftarrow CH_j = \{\} \cup w_{jt}\} \)
The procedure adopted in this algorithm 6.9.1 is to classify the nodes for a given cluster which consists of a subset of nodes. The pos value is the same for all the nodes, the \((N - \text{Tuple}_i)_{N+1} = ((Co - ordv_i)_1, (Co - ordv_i)_2, \ldots, (Co - ordv_i)_N, \text{pos}), C_j \{n_1, n_2, n_3, n_k\}, n_k \in C_j, 1 \leq k \leq k, 1 \leq j \leq N\). The vertex with \((Co - ordv_i)_j = 0\) is treated as a cluster head of that cluster \(C_j\{\}\) and the vertex with \((Co - ordv_i)_j \neq 0\) is treated as a member node of that cluster \(C_j\{\}\). The Figure shows the algorithm 6.5. The above algorithm 6.9.1 shows the construction of clusters by using LMUAC algorithm 6.7.1.

![Figure 6.5: Construction of Clusters(LMUAC)](image)

### 6.10 Identification of Gateway nodes in a network

Gateway nodes are nodes in a non-clusterhead state located at the periphery of a cluster. These types of nodes are called gateways because they are able to listen to transmissions from another node which is in a different cluster \[72\]. To accomplish this, a gateway node must have at least one neighbor that is
Development of a Clustered Network

a gateway of another cluster \[75\]. The following algorithm is used to identify the gateway nodes in a network. The \(d(u_j, v_j) = 1\) and the position value of \(v_j\) and \(u_j\) is different then they are treated as a gateway node in a network. The algorithm 6.10.1 takes care of identifying gateway nodes in a network. For the algorithm 6.11.1 shows the identification of gateway nodes in the considered example.

6.10.1 Algorithm for Identification of Gateway nodes

1: \(\text{Gateway Node} = \phi = GW\{\}\)
2: \(\text{NonGatewayNode} = \phi = NGW\{\}\)
3: **Step 0:** Set \(p = (p_1, p_2, p_3, p_q), q \text{ is number of edges in graph } G\)
4: \(p_j = (u_j, v_j), \text{where } d(u_j, v_j) = 1\).
5: **Step 1:** Input \((N\text{Tuple } v_i)_{N+1} = ((Co-ord v_i)_1, (Co-ord v_i)_2, \ldots, (Co-ord v_i)_N, pos)\)
6: \(1 \leq i \leq n, 1 \leq j \leq N\)
7: **Step 2:** Identify gateway nodes.
8: Consider \(p_k\)
9: FOR \(1 \leq k \leq q, p_j = (u_k, v_k)\)
10: if \((\text{pos}_u_k \neq \text{pos}_v_k)\)
11: \(\{GW\} \leftarrow GW\{\} \cup u_k, v_k, NGW\{\} \leftarrow NGW\{\}\)
12: ELSE
13: \(\{NGW\} \leftarrow NGW\{\} \cup u_k, v_k, GW\{\} \leftarrow GW\{\}\)
14: **Step 3:** Exit, Output
15: \(GW\{\text{nonemptyset}\}\)
16: \(NGW\{\text{emptyset or nonemptyset}\}\)

6.11 Hierarchical address for the cluster heads in clustered network

Hybrid routing provides routing through the implementation of a hierarchical approach \[76\]. In a hierarchical approach the network is organized into subsets of nodes, known as clusters. This topology organization reduces network traffic because a node only needs to have knowledge of the routing information within its cluster and not of the entire network. Hybrid routing, also known as cluster-
Development of a Clustered Network

Figure 6.6: Identification of Gateway nodes in a Network (LMUAC)

based routing is a convenient scheme for developing efficient routing algorithms in MANET. Apart from making a large network appear smaller, one significant feature of cluster-based routing is that it can make a dynamic topology appear less dynamic \[70\]. In order to implement a dynamic hybrid routing scheme, efficient clustering algorithms must be designed. After clustering, each node in the hierarchy can be assigned a hierarchical address that indicates its position in each level of the hierarchy. Routing can easily be carried out using such addresses. The example from \[68\] to explain this. A n-node network with three hierarchy levels is created by recursive clustering. The terms Level-0 to refer to the original network, Level-1 to refer to the structure obtained by clustering once, and Level-2 to refer to the structure obtained by clustering Level-1. Each node in the network can be assigned a n-level hierarchical address.

The input is a $CH_j\{}$ which is a non empty set and it contains only one vertex in it. Now our aim is to generate the hierarchical code for the cluster heads of that cluster. The process of generating the unique code at the level 1 is given by for a given $CH_j\{} = (Co – ordv_k) \in 0 \rightarrow 0$ and $(Co – ordv_k), \neq 0 \rightarrow 1, i \neq j$ the length of the tuple is $N, 1 \leq i \leq N$. The algorithm 6.11.1 is used to generate the unique code for the cluster head at level 1 hierarchy. The Figure 6.7 shows the hierarchical code for the example.
6.11.1 Algorithm for Generating Unique code for CH at
1 level hierarchy (LMUAC)

1: \( HCH_j = \{\} = \phi \), empty set
2: Step 0: Input \( CH_j = \{w_{j1}\} \)
3: \( N \) - Tuple \( w_{jt} = ((Co - ord w_{j1}), (Co - ord w_{j2}), \ldots (Co - ord w_{jN})) \)
4: Step 1: FOR \( 1 \leq j \leq N \)
5: FOR \( 1 \leq i \leq N \)
6: IF \((Co - ord w_{jt})_i == 0)\)
7: \( \{HCH_j\} \leftarrow HCH_j \cup \{0\} \)
8: ELSE
9: \( \{HCH_j\} \leftarrow HCH_j \cup \{1\} \)
10: EXIT
11: Step 2: Output
12: \( HCH_j \{\text{combinations of } 1 \text{ and } 0 \text{ of length } N\} \)

Figure 6.7: Generating unique code for CH at 1 Level hierarchy (LMUAC)

6.12 Expansion of existing network (Adding a new node):

Let \( G = (V, E) \) be the graph representation of the existing network. Let \( C_1, C_2, C_3, \ldots, C_N \) be the \( N \) clusters of \( V(G) \) and \( v_1, v_2, \ldots, v_N \) be the corresponding cluster heads. The possibility of adding a new vertex to any vertex of any cluster through one edge is explored. It is done so that there will be no
change in the existing clusters (except that the corresponding clusters to which, the adding of a new vertex has one more edge and one more vertex) and no change in cluster heads. We first state and prove a theorem and then give an algorithm to check whether a vertex can be added to the given vertex of the given network.

**Theorem 6.6.** Let $G(V, E)$ be the graph representation of a given network. Let $C_1, C_2, C_3, \ldots, C_N$ be the clusters and $v_1, v_2, \ldots, v_N$ be the corresponding cluster heads. A new vertex $u$ can be added to a vertex $v$ in the cluster $C_j$, $1 \leq j \leq N$ through a single edge if and only if the unique clustering ID generated to the vertex $u$ is not the same as unique clustering ID of any vertex in $V_d$ if exists, where $V_0, V_1, V_2, \ldots, V_d, \ldots, V_e(v_j)$ is the distance partition of $V(C_j)$ with respect to the vertex $v_j$ and $d(v_j, u) = d$.

**Proof.** As the $(Co - ordN - Tuple_u)_j$ is the unique clustering ID of $u$ it cannot be same as unique clustering ID of any vertex in any cluster $C_i, i \neq j$. In $C_j$, all the vertices $w$ are at distance $d$ from the cluster head $v_j$ hence the same value of $(Co - ordN - Tuple_w)_j$ as $(Co - ordN - Tuple_u)_j$ because $d(v_j, u)$. If for some such vertex $x$, all the $Co - ordN - Tuple$ in the unique clustering ID are same as corresponding $Co - ordN - Tuple$ of $u$ in its unique clustering ID, then the vertices $x$ and $u$ are not strongly resolved by any of the cluster heads, then $u$ cannot be added through a single edge. If the above is not the case then $u$ can be added to $C_j$, through a single edge. \qed

Example: Addition of a vertex $x/\dot{x}$.

Case(i) If $d(u, CH_1) > e(CH_1)$.

Consider the Figure 6.5. Let $C_1$ be a cluster and $e(CH_1) = 3$ as shown in Figure 6.5. Let $x$ be a new vertex to be added to the cluster $c_1$ in the existing network through a single edge. Let $x$ be a vertex which is at a distance greater than the $e(CH_1)$ with respect to $CH_1$. From the Figure the $e(CH_1) = 3$. Let $u$ be the only vertex which is at distance 3 from $CH_1$ from the Figure. The
new vertex to be added to the existing cluster $C_1$, which is at a distance 4. The
distance of vertex $x$ is greater than the $e(CH_1)$, then the new vertex should be
adjacent to the only vertex $u$ which is at distance three from the $CH_1$. The
$N-Tuple$ of $u$ from the Figure is (3875). The new vertex which is adjacent to
the vertex $u$ will have the $(N-TupleCo-ord)$ of (4986). The new $(Co-ord)$
of $x$ generated is by incrementing $N-Tuple$ of $u$ by 1. The newly generated
$N-Tuple$ of $x$ is strongly resolved by $CH_1$ and $x \in C_1$. The newly added
vertex which is at 4 hop distance from the cluster head which is $> e(CH_1)$.
Then add this new vertex through the single edge. The new vertex $x$ to be
added to the existing cluster without modifying the cluster head numbers and
cluster structure is verified. Consider the distance partition of the cluster $C_1$
with respect to the cluster head $L_1$, parallel consider the new vertex to be
added to the existing cluster $C_1$ with respect to cluster head $L_1$ separately.
Find the intersection of $L_1 \cap x$. From the above theorem 6.6 if $L_1 \cap x = 0$. Then
add the new vertex to the cluster $C_1$ without any modification of the cluster
structure. Update the $e(CH_1)$ and adjacent vertex to which the new vertex
added through the single edge neighbor member Table. The Table 6.12 shows
the update structure.

Distance partition with respect to $L_1$ with vertex $x$.

Case (ii) If $d(\acute{x},CH_1) < e(CH_1)$.

Table 6.3: Distance partition with respect to $L_1$ with vertex $x \geq e(C_j)$.

<table>
<thead>
<tr>
<th>$d(CH_1)$</th>
<th>$L_1$</th>
<th>$x$</th>
<th>$L_1 \cap x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1545),(1764)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>(2434)(2653)(2764)(2875)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>(3875)(3545)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>(4986)</td>
<td>-</td>
</tr>
</tbody>
</table>

Let $\acute{x}$ be a new vertex which is at a distance $d$ from cluster head $L_1$. Let
Development of a Clustered Network

$d < e(CH_1)$, then the new vertex $\hat{x}$ should be adjacent to one of the vertices in the cluster $C_1$. Let $v$ be a vertex, which is at 2 hops from cluster head $L_1$ which is less than the eccentricity of the $CH_1$. From the Figure let $v$ have $N_{Tuple}$ value (2764). Let $\hat{x}$ be adjacent to vertex $v$, then the new $Co - ord$ of $\hat{x}$ generated is by incrementing $N_{Tuple}$ of $v$ by 1. The newly generated $N - Tuple$ of $\hat{x}$ is (3875) which is strongly resolved by $CH_1$ and $\hat{x} \in C_1$. The newly added vertex which is at 3 hop distance from the cluster head which is $\leq e(CH_1)$. Then add this new vertex through the single edge. The new vertex $\hat{x}$ to be added to the existing cluster without modifying the cluster head numbers and cluster structure is to be verified. Consider the distance partition of the cluster $c_1$ with respect to the cluster head $l_1$, parallel consider the new vertex $\hat{x}$ to be added to the existing cluster $C_1$ with respect to cluster head $L_1$ separately. Find the intersection of $L_1 \cap \hat{x}$. From the above theorem if $L_1 \cap \hat{x} = 1$. Then the addition of a new vertex to the cluster $C_1$ is not permitted. The Table 6.12 shows the update structure. The example shows the addition or deletion of a node in a cluster which is shown the Figure 6.8

Table 6.4: Distance partition with respect to $L_1$ with vertex $\hat{x} \leq e(C_j)$.

<table>
<thead>
<tr>
<th>$d(CH_1)$</th>
<th>$L_1, CH_1$</th>
<th>$\hat{x}$</th>
<th>$L_1 \cap \hat{x}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1545),(1764)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>(2434),(2653),(2764),(2875)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>(3875)(3545)</td>
<td>(3875)</td>
<td>1</td>
</tr>
</tbody>
</table>

6.13 Conclusion

In this chapter, Land Marks for Unique Addressing (LMUA) algorithm is developed to generate unique ID for each and every node which leads to the formation of overlapping/Non overlapping clusters based on unique ID. To overcome
the draw back of the developed LMUA algorithm, the concept of clustering is introduced. Based on the clustering concept a Land Marks for Unique Addressing and Clustering (LMUAC) Algorithm is developed to construct strictly non-overlapping clusters and classify those nodes into Cluster Heads, Member Nodes, Gateway nodes and generating the Hierarchical code for the cluster heads to operate in the level one hierarchy for wireless communication switching. The expansion of the existing network can be performed or not without modifying the cost of adding the clusterhead is shown. The developed algorithm shows one way of efficiently constructing the multi-hop non-overlapping clusters based on developed Unique ID.