Study of different mobility models and clustering algorithms like Weighted Clustering Algorithm (WCA) and Dynamic Mobility Adaptive Clustering Algorithm (DMAC)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

SURYADEEP BISWAL

DIMOL SOREN

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2007

Study of different mobility models and clustering algorithms like Weighted Clustering Algorithm (WCA) and Dynamic Mobility Adaptive Clustering Algorithm (DMAC)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

SURYADEEP BISWAL

DIMOL SOREN

Under the Guidance of

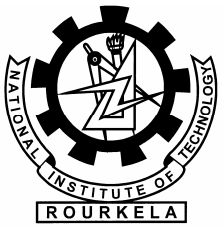
Prof. S.K. Rath

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2007



National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "STUDY OF DIFFERENT MOBILITY MODELS AND CLUSTERING ALGORITHMS LIKE WEIGHTED CLUSTERING ALGORITHM (WCA) AND DYNAMIC MOBLITY ADAPTIVE CLUSTERING ALGORITHM (DMAC)" submitted by Sri Suryadeep Biswal and Dimol Soren in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. S.K. Rath
Dept .of Computer Science and Engineering
National Institute of Technology
Rourkela - 769008

Acknowledgement

We express our sincere gratitude to Prof. S.K. Rath, Department of Computer Science and Engineering, National Institute of Technology, Rourkela, for his valuable guidance and timely suggestions during the entire duration of our project work, without which this work would not have been possible.

We would also like to convey our deep regards to all other faculty members and staff of Department of Computer Science and Engineering, NIT Rourkela, who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on our part to finish this project work.

Dimol Soren
Roll No: 10306009

Suryadeep Biswal
Roll No: 10306017

CONTENTS

ABSTRACT

This project addresses issues pertaining to mobile multi-hop radio networks called mobile ad hoc networks (MANET), which plays a critical role in places where a wired backbone is neither available nor economical to deploy. Our objective was to form and maintain clusters for efficient routing, scalability and energy utilization. To map the cellular architecture into the mobile ad hoc network cluster heads are elected that form the virtual backbone for packet transmission. However, the constant movement of the nodes changes the topology of the network, which perturbs the transmission. This demands the cluster maintenance.

Weighed Clustering Algorithm (WCA)[4] and Distributed and Mobility adaptive Clustering (DMAC) [1,2,3] are two better proven algorithms on which we have implemented different mobility models like Random Walk (RW), Random Way Point (RWP) and Random Direction (RD). In both the algorithms each node is assigned some weight .In WCA the weight is a function of parameters like Battery power, mobility, transmission range and degree of connectivity. DMAC is mobility adaptive, i.e. it takes the mobility of the nodes into consideration while forming the clusters. We have chosen some measuring parameters like no of clusterheads, Average cluster lifetime, and Reaffilation rate for comparing the performance of both the algorithms.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Introduction

The emergence of powerful hand-held devices like cell phones, pagers coupled with the advancement of wireless communication systems have paved the way for a variety of mobile computing and wireless networking technologies recently. The real boon for wireless networks is its ability to support user mobility. Mobility brings a new dimension to problem solving in this domain, resulting in an unpredictable resource requirements and uncertainty in network connectivity. In this paper we assume our wireless network to be an *ad-hoc* network[4] i.e. networks without any fixed infrastructure. They play a critical role where a wired (central) backbone is neither available nor economical to build such as law enforcement options, disaster recovery situations etc. These situations demand a network where all the nodes including the base stations are potentially mobile. Hence we also assume that all the algorithms are distributive and not central because of the absence of any central station. The movement of the nodes is purely random in this network and doesn't depend upon the movement of any other node.

To manage the nodes and communication among them it is necessary to study their movement and also group them into different clusters. Each cluster is represented by a *clusterhead* that is selected by a function of several parameters. A calculated weight is assigned to each node and then different selection techniques are applied to select the clusterhead. The nodes in different clusters communicate through their clusterheads. Basically three different mobility models like Random Way Point (RWP), random Walk (RW) and Random Direction (RD) [7, 8] are used to simulate the nodes movements. The movements of different nodes vary in all of the three models. In RWP model the nodes select random speeds and directions and start moving in that direction towards a randomly selected destination. Upon reaching there they pause for a certain period (*pause time*) and again start moving. RW model is similar to RWP model except in this model the pause time is zero. In random direction model the nodes change their direction and speed upon reaching the boundaries of the simulation area. Different clustering algorithms such as Weight based Clustering Algorithm (WCA) [4, 5, 6] and Distributive and Mobility Adaptive Clustering Algorithm (DMAC) [1, 2, 3] are used to partition the nodes into clusters and manage them.

CHAPTER 2

DIFFERENT MOBILTY MODELS

Random Direction Model (RD)
Random Way Point Model (RWP)
Random Walk Model (RW)

In order to simulate a new protocol for an ad-hoc network, it is imperative to use a mobility model that accurately represents the mobile nodes (MNs) that will eventually utilize the given protocol. It should mimic the movements of real MNs. Changes in speed and direction must occur and in reasonable time slots.

Three different mobility models were described by Albert Einstein in 1926. Those are [7,8]:

1. Random Direction (RD)
2. Random Waypoint (RWP)
3. Random Walk (RW)

2.1 RANDOM DIRECTION MODEL (RD):

The Random Direction mobility model was created to overcome density waves in the average number neighbors produced by the Random Waypoint model. A density wave is the clustering of nodes in one part simulation area. In the case of the Random Waypoint mobility model, this clustering occurs near the centre simulation area. In the Random Waypoint model, the probability of an MN choosing a new destination located in the centre of the simulation area, or a destination which requires travel through the middle of simulation area is high. Thus, the MNs appear to coverage, disperse and coverage.

In order to alleviate this type of behavior and promote a semi-constant number of neighbors throughout the evaluation, the Random Direction mobility model is used. An MN then travels to the border of the simulation area in that direction. Once the simulation boundary is reached, the MN pauses for a specified time, chooses an angular direction and continues the process. MNs usually pause at the border of the simulation area; so the average hop count for the data packets at the Random Direction Mobility model will be much higher than the average hop count of most other mobility models. Network partitions will be more likely with the Random Direction mobility model compared to other mobility models. A slight modification to the Random Direction mobility model is that, in modified Random Direction Model [8] MNs continue to

choose random directions but they are no longer forced to travel to the simulation boundary before stopping to change direction. Instead, an MN chooses a random direction and a destination anywhere along that direction of travel. The MN then pauses at this destination before choosing Random Direction. This modification to the Random Direction Mobility model produces a movement pattern that can also be simulated by RW with pause time.

2.2 RANDOM WAYPOINT (RWP):

Random Waypoint mobility models include pause time between changes in direction and/or speed. MN begins by staying in one location for a certain period of time. Once this time expires MN chooses a random destination between [min, maxspeed]. The MN then travels towards the newly chosen destination at the selected speed upon arrival; it pauses for a specified time period before starting the process again.In most of the performance investigation that uses the random waypoint mobility model the MNs are distributed randomly around the simulation area. The average MN neighbor percentage is the cumulative percentage of total MNs that are given MNs in the network and if a node has 10 neighbors in a network of 200 nodes, then the nodes current neighbor percentage is 20%. A neighbor of an MN is a node within the MNs transmission range.

In the following, we present three possible solutions to avoid this initialization problem. First, save the locations of the MNs after a simulation has executed long enough to be past this initial high variability, and use this position file as the initial starting point of the MNs in all future simulations. Second, initially distribute the MNs in a triangle distribution may distribute nodes in the Random Waypoint mobility model more accurately than initially placing the MNs randomly in the simulation area. Lastly, discard the initial 1000 seconds of simulation trial. Discarding the initial 1000 seconds of simulation time has an added benefit over the solution proposed.There is also a complex relationship between node speed and pause time in the Random Waypoint Mobility model. For example, a scenario with fast MNs and long pause time actually produces a more stable network than a scenario with slower MNs and shorter pause times. If the Random Waypoint model is used in a performance evaluation, appropriate parameters need to be evaluated.

2.3 RANDOM WALK (RW):

Since many entity in nature move in extremely unpredictable way,the Random Walk mobility model was developed to mimic the erratic movement.In this mobility model,an MN moves from its current location to a new location by Randomly choosing a direction and speed in which to travel.The new speed and direction are both chosen from pre-defined ranges [speedmin,speedmax] and [0,2∏] respectively.Each movement in the Random Walk mobility model occur in either a constant time interval 't' or a constant distance traveled ,at the end of which a new direction and speed are calculated.If an MN which moves accordingly to this model reaches a simulation boundary, it bounces off simulation border with an angle determined by the incoming angle. Many derivatives of Random Walk Mobility model have been developed including the 1-D, 2-D, 3-D,…., d-D.The 2-D random walk mobility model is of special interest.

The Random Walk mobility model suffers from "*Central Effect"*[7] i.e. in this model the nodes tend to cluster at the center of the simulation area. Hence the mobility of each node decreases over time. Hence the average speed of all the nodes decreases over time. But this model has a no. of advantages over the other two models such as (1) Here the movement of the nodes follow a certain pattern and hence this model is suitable for most of the simulations. (2)Here the speed variations of the nodes are not that rapid.(3) Moreover it captures the best features of both RWP and RD.

An Example

In the following examples we describe the probable movement of one node under one of the above three models:

Random Walk

MN begins it movement in the centre of each the 300*600 simulation area. At the each point, the MN randomly chooses a direction between 0 and 2∏ and a speed between 0 and 10m/sec.The MN is allowed to travel for 60 sec between changing direction and speed or MN is allowed to move 10 steps between changing direction and speed.

This is a memory less mobility pattern because it retains no knowledge concerning past location and speed.

Random Waypoint

MN starts its movement by choosing a new location in the 300*600 simulation area; it chooses a speed that is uniformly distributed between 0 and 10 m/sec. The MN is allowed to travel for 60 sec before choosing another location, between the changes of direction it halts for a certain pause time. Random waypoint mobility model is similar to Random Walk if the pause time is zero.

Random Direction

MN begins its movement by choosing a speed and a direction in the 300*600 simulation area, moves in that direction until it reaches the boundary, upon reaching it pauses for certain period of time. And before starting again it chooses Angular direction between 0 and 180.In this MNs actually removes the clustering of nodes that was observed in Random Walk.

CHAPTER 3

A WEIGHTED CLUSTERING ALORITHM (WCA)

In this chapter we discuss a weight based clustering algorithm (WCA)[4,5,6] that takes into consideration the no of nodes a clusterhead can handle ideally (without any severe degradation of performance), mobility, battery power and transmission power of the nodes. Unlike other algorithms which are invoked periodically resulting in high communication overhead, our algorithm is adaptively invoked based on the mobility of the nodes. The clusterhead election procedure is delayed as long as possible to reduce the computation cost. It also achieves *load balancing* [4] among the nodes by specifying an upper bound on the no of nodes a clusterhead can handle ideally. it also achieves *connectivity* [4] i.e. all the nodes are connected to each other.

Algorithm WCA:

3.1Preliminaries:

The network formed by the clusters can be represented by an undirected graph **G=(V,E)** ,where V represents the set of nodes v_i and E represents the set of links e_i. We assume that |V| is constant but |E| changes with creation and deletion of links. Clustering can be thought of a graph-portioning problem with some given constraints, but it is a NP hard problem. More formally we look for a set of vertices S Є V(G), such that

$$\grave{U}_{v \in S} \; N[v] = V(G)$$

Here $N[v]$ is the neighborhood of *v*, defined as:

$$N[v] = \grave{U}_{v' \in V, v' \neq v} \{ v' | dist(v,v') < tx_{range} \}$$

Where tx_{range} is the transmission range of node V. The set S is called a *Dominating Set* [4] if every vertex of V either belongs to S or has a neighbor in S. In other words the domination set of a graph is the set of its clusterheads. When the set of clusterheads change it is called a *Dominating set update*.

Design Philosophy:

Choosing an optimal no of clusterheads that will yield high throughput but incur as low latency as possible is still an important problem. This algorithm makes use of a *combined weight metric*

[4] which takes into account several system parameters like the ideal node-degree, transmission power, battery power and mobility of the nodes. This results in a fully distributed system where all the nodes in the network share the same responsibility and act as clusterheads. However, more clusterheads results in an extra number of hops for a packet, when it gets routed from source to destination. Thus, this solution leads to higher latency, more power consuming and more information processing per node.

On the other hand, to maximize the resource utilization, we can choose to have the minimum no. of clusterheads to cover the whole geographical area over which the nodes are distributed. The whole area can split up into zones, the size of which can be determined by the transmission range of the nodes. This puts a lower bound on the number of clusterheads required. Ideally, to reach this lower bound, the uniform distribution of the nodes is necessary over the entire area. Also, the total no. of nodes per unit area should be restricted so that the clusterhead in a zone can handle all the nodes therein. However, such zone-based clustering is not a viable solution due to the following reasons. The clusterheads would typically be centrally located in the zone; if they move new clusterheads has to be selected. It might so happen that none of the other zone in that zone is centrally located. Therefore, to find a new node that act as a clusterhead with the other nodes in its transmission ranges might difficult. Another problem arises due to non-uniform distribution of nodes over the whole area. If a certain zone becomes densely populate due to migration of nodes from other zones, then clusterhead might not be able to handle all the traffic generated by the nodes because there is inherent limitation of number of nodes a clusterhead can handle. We propose to elect the minimum number clusterheads which can support all the nodes in the system satisfying the above constraints.

3.2The Basis for the Algorithm:

To evaluate a candidate clusterhead, we take into consideration its degree, transmission power, mobility and battery power. The following features are considered in our clustering algorithm:

- The clusterhead procedure is not periodic and is invoked as rarely as possible. This reduces system updates and communication and computation costs. The clustering algorithm is not invoked unless the relative distances between the node and the clusterheads change.

- Each clusterhead can ideally support δ (a predefined threshold) nodes to ensure medium
- access control (MAC) functioning. If the clusterhead tries to serve more nodes than it is capable of, the system efficiency suffers in the sense that nodes will incur more delay because they have to wait longer for their turn to get their share of resource. A high system throughput can be achieved by limiting or optimizing the degree of each clusterhead.
- The battery power can be efficiently used within certain transmission range, i.e., it will take less power for a node to communicate with other nodes if they are within the close distance of each other. A clusterhead consumes more battery power than the ordinary nodes.
- Mobility is an important factor in deciding the clusterhead. In order to avoid the frequent change of clusterhead, it is desirable to elect a clusterhead that does not move quickly. When the clusterhead moves fast, the nodes may be detached from the clusterhead and as a result, a reaffilation occurs. Reaffilation takes place when one of the ordinary nodes moves out of a cluster and joins another existing cluster.
- A clusterhead is able to communicate better to its neighbors having closer distances within the transmission range. As the nodes move away from the clusterhead, the communication may become difficult mainly due to signal attenuation with increasing distance.

3.3Details of WCA:

Based on the previous discussions, a weight based clustering algorithm (WCA) that effectively combines each of the above system parameters with certain weighing factors chosen according to the system needs. For example power control is very important in CDMA networks, thus the weight of the corresponding parameter can be made larger. The flexibility of changing the weight factors helps us apply our algorithm to various networks. The output of the clusterhead election procedure is a set of nodes called *dominating* Set [4]. The clusterhead election procedure is invoked at the time of system activation and when the current dominant set is unable to cover all the nodes. An invocation of the election algorithm does not necessarily mean that all the clusterheads in the previous dominant set are replaced by the new ones. if a node detaches itself from its current clusterhead and attaches to another clusterhead, then the involved clusterheads update their member list instead of invoking the election algorithm.

3.4Clusterhead Election Procedure:

(1) Find the neighbors N(v) of each node 'v' which defines its degree d_v , as

$$\mathbf{d_v = |N(v)| = \sum\{ dist(v,v') < trange \}} \qquad \mathbf{V' \epsilon V,\ v' \neq v}$$

(2) Compute the degree difference, $\mathbf{\Delta_v = | d_v - \delta |}$, for every node v, where $\delta = \max^m$ possible degree for a node.

(3) For every node compute the sum of the distances, D_v, with all its neighbors as:

$$\mathbf{D_v = \sum \{dist\ (v, v')\}} \qquad \mathbf{v' \epsilon N\ (v)}$$

(4) Compute the running average of speed for every node till the current time 'T' .This gives a measure of Mobility and is denoted by $\mathbf{M_v}$.

$$\mathbf{M_v = \left\{ \sum_{t=1}^{T} \sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2} \right\} / T}$$

Where (X_t, Y_t) and (X_{t-1}, Y_{t-1}) are the co-ordinates of node 'v' at time't' and't-1' respectively.

(5) Compute the cumulative time, Pv, for which a node acts as a clusterhead. P_v indicates how much battery power has been consumed.

(6)Finally Calculate the Wv for each node 'v' where

$$W_v = w_1 \Delta_v + w_2 D_v + w_3 M_v + w_4 P_v .$$

Where w_1, $w2$, w_3, w_4 are the weighing factors for the system parameters.

(7) Choose the node with the smallest Wv, as the clusterhead. None of the neighbors of the chosen clusterhead are allowed to participate in the election procedure.

(8) Repeat this step till every node is either a clusterhead or member of some cluster.

3.5An Example:

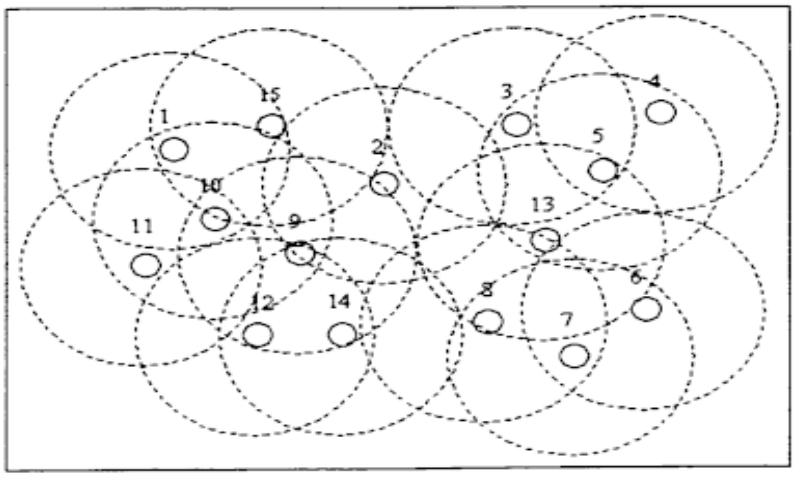


Figure 3.4. Initial configuration of nodes.

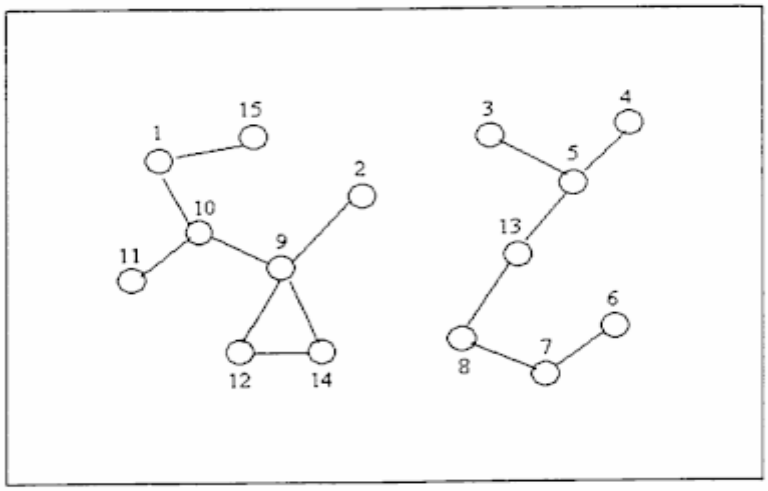


Figure 3.5. Neighbors identified.

Figure 3.4 shows the initial configuration of the nodes in the network along with their nodeIDs.The dotted circles with equal radius represent the fixed transmission range for each node. In the figure 3.5 neighbors of each node (nodes within the transmission range of a given node) are identified.

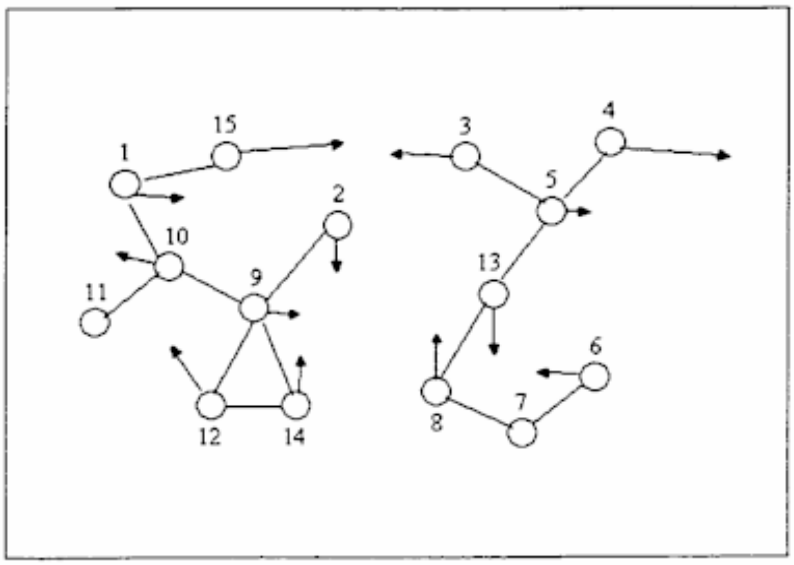


Figure 3.6. Velocity of the nodes.

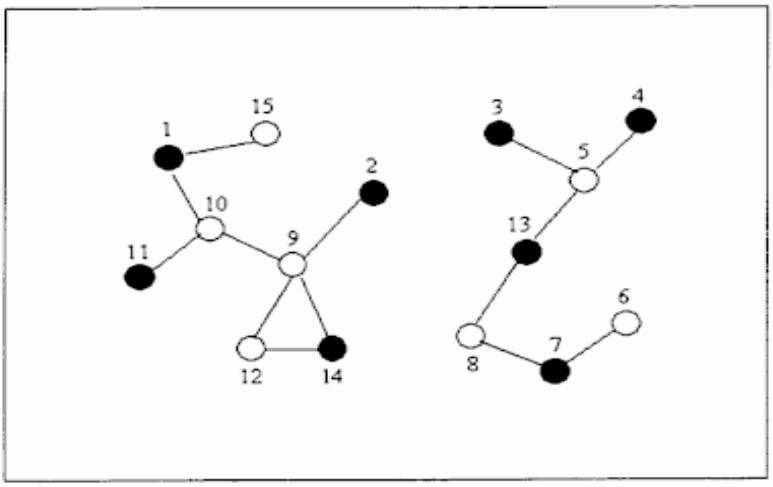


Figure 3.7. Clusterheads identified.

Figure 3.6 shows the mobility i.e. velocity of different nodes. This describes the random movement of the nodes. In the figure 3.7 the clusterheads are identified by invoking WCA.

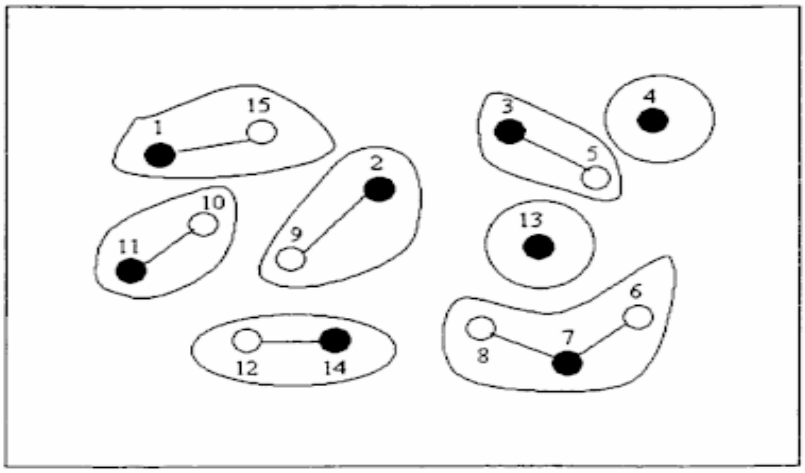


Figure 3.8. Clusters identified.

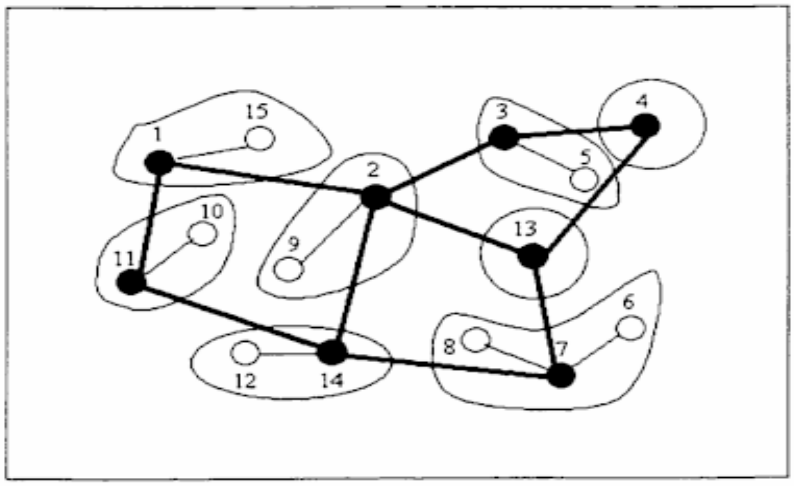


Figure 3.9. Connectivity achieved.

Figure 3.8 shows how the clusters are formed .Similarly the figure 3.9 shows how connectivity is achieved among different nodes i.e. a data packet can be routed from a node to any other node. The table 3.1 given below explains the execution of WCA .Different parameters are explained above. We have taken the weight as w_1=0.7, w_2= 0.2, w_3=0.05, w_4=0.05 such that their sum is 1.Values of some of the parameters are chosen randomly.

Table 3.1. Execution of WCA

Node id	d_v Step 1	Δ_v Step 2	D_v Step 3	M_v Step 4	P_v Step 5	W_v Step 6
1	2	0	6	2	1	1.35
2	1	1	4	2	2	1.70
3	1	1	3	3	1	1.50
4	1	1	3	4	2	1.60
5	3	1	9	1	4	2.75
6	1	1	3	2	2	1.50
7	2	0	6	0	0	1.20
8	2	0	7	3	3	1.70
9	4	2	13	2	6	4.40
10	3	1	12	2	7	3.55
11	1	1	3	0	1	1.35
12	2	0	5	3	4	1.35
13	2	0	7	3	2	1.65
14	2	0	5	2	0	1.10
15	1	1	3	4	3	1.65

3.6Load Balancing:

The load handled by a cluster depends on the no of nodes supported by it. A parameter called *Load Balancing Factor* (LBF)[4] measures how well balanced the clusterheads are. As the load of any clusterhead can be represented by the cardinality of its cluster size, LBF can be defined as:

$LBF = N_c / \{\sum_i (N_i - \mu)2$

Where N_c = Number of clusterheads.

N_i = Cardinality of clusterhead i.

$\mu = (N – N_c)/ N_c$, is the average no of neighbors of a clusterhead .

N= Total no of nodes.

- A high value of LBF indicates better load distribution and for a perfectly balanced system it tends to '∞'.

3.7Simulation Study:

We simulated a system with no of nodes(N) € (10,200), Transmission range =10, Maximum displacement (mobility) =2 and in a 100*100 grid. We assumed δ=10, max disp=10, tRange €[0-60], w1=0.7 , w2=0.2, w3=0.05, w4=0.05 . (w1 was high to keep the node degree as close to the ideal as possible). To measure the performance of the system three metrics were identified.

(1) Average no of Clusters: This indicates the average no of clusters or clusterheads in the system. Every time a dominant set is identified its cardinality gives the no of clusterheads.

(2) Number of Dominant Set updates: The set of clusterheads at any given time forms the dominant set of a system. How frequently the clusterheads are being updated affects the performance. The dominant set update takes place when a node can no longer be a member of any of the existing clusterheads.

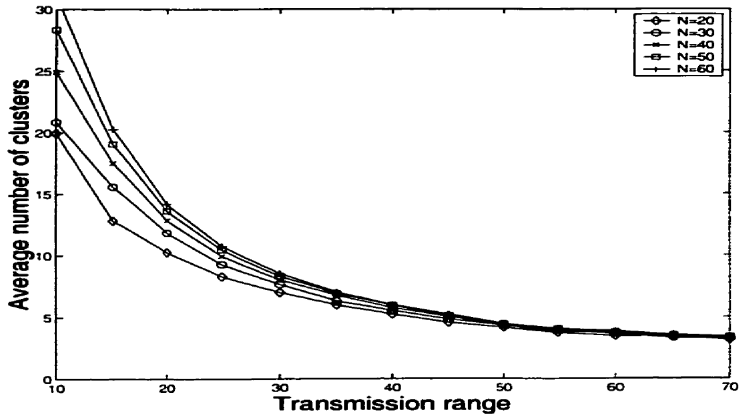
(3) Reaffilations per unit time: How many nodes are leaving a clusterhead and joining another also affects the system. Reaffilation count is incremented when a node gets dissociated from the current cluster and joins another cluster within the current *Dominant Set.*

The above parameters were plotted against transmission range that was varied between 0 and 70.No of nodes were varied simultaneously from 20 to 60.

The Results we obtained are:

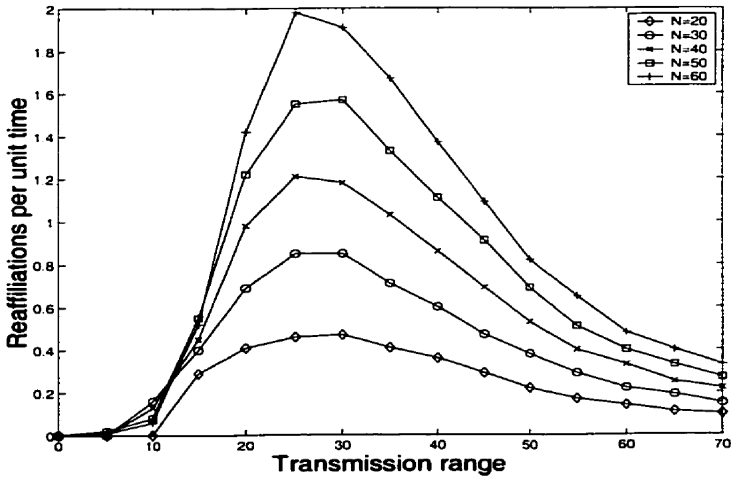
- **Average No of clusters Vs Transmission range:**

As the transmission range of the nodes increase, average no of clusters tend to decrease because more no of nodes come within the transmission range of one cluster.



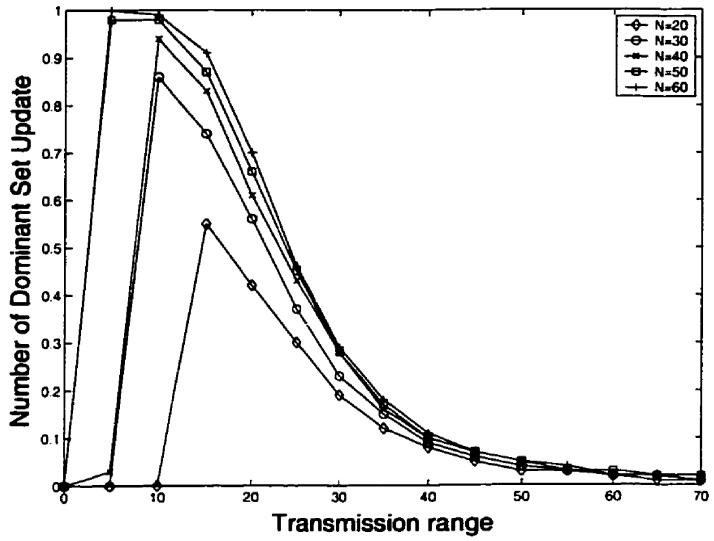
- **Reaffilations per unit time Vs Transmission range:**

In the beginning as the transmission range increases the reaffilation rate increases since more no of nodes leave their clusters to join other clusters but later as the transmission range becomes very high the reaffilation rate decreases.



- **No of Dominant Set Updates Vs transmission range:**

Initially as the transmission range increases there is a rapid change in the dominant set as observed from the graph, but as the transmission range further increases the dominant set updates decrease since the set is able to cover all the nodes in the network.



CHAPTER 4

DISTRIBUTIVE AND MOBILITY ADAPTIVE CLUSTERING (DMAC)

DMAC(Distributed and Mobility-adaptive Clustering):

In DMAC [1, 2, 3] we do not assume that during the clustering process the nodes of the network need not to move. This makes this algorithm suitable for both the clustering set up and its maintenance. Adaptation to changes in the network topology is now made possible by letting each node to properly "react" not only to the reception of a message from other nodes, but also to the failure of a link with another node or to the presence of a new link.

In the description of the procedures of our Distributed and Mobility-Adaptive Clustering (DMAC algorithm), we still assume that a message sent by a node is received correctly within a finite time (a step) by all its neighbors. We also assume that each node knows its own ID, its weight, its role (if it has decided to be a clusterhead or an ordinary node) and the ID, the weight and the role of all its neighbors (if they have already decided their role). When a node has not yet decided what its role is going to be, it is considered as an ordinary node.

4.1Algorithm:

Except for the procedure that each node executes as soon as it starts the clustering operations, the algorithm is message driven. Here we use the two types of messages

- CH(v)
- JOIN(v,u)

Furthermore we assume that:-

- Every node is made aware of the failure of a link, or the presence of a new link by a service of a lower level (this will trigger the execution of the corresponding procedure);
- The procedures of DMAC (M-procedures, for short) are "atomic",i.e., they are not interruptible;
- At the clustering set up or when a node is added to the network its variables *Clusterhead*, Ch(-), and $_{Cluster}$(-) are initialized to nil, false and Ø, respectively.
- ┌(x) represents the set of neighboring nodes of any node x.

The following is the description of the six M-procedures as executed at each node v.

- ***Init***. At the clustering set up, or when a node v is added to the network, it executes the procedure *Init* in order to determine its role. If among its neighbors there is at least a clusterhead with bigger weight, then v will join it. Otherwise it will be a clusterhead. Notice that a neighbor with a bigger weight that has not decided its role yet(this may happen at the clustering setup, or when two or more nodes are added to the network at the same time), will eventually send a message(Every node executes *Init* procedure). If this message is a CH message, then v will affiliate with the new cluster head.

Procedure Init;

```
Begin
  If{z Є ┌(v):w_z> w_v /\ Ch(v)}≠ Ø
    Then begin
          x:=maxw_z > w_v {z:Ch(z)};
      send JOIN(v,x);



          clusterhead:=x
```

```
            end
        else begin
            send CH(v);
            Ch(v):=true;
            Clusterhead:=v;
            Cluster(v):={v}
             End
end;
```

- ***Link_failure.*** Whenever made aware of the failure of the link with a node u, node v checks if its own role is clusterhead and if u used to belong to its cluster. If this is case, v removes u from Cluster(v). If v is an ordinary node, and u was its own clusterhead, then it is necessary to determine a new role for v. To this aim, v checks if there exists at least a clusterhead z $\in$ Γ(v)such that $w_z > w_v$. If this is the case, then v joins the clusterhead with the bigger weight, otherwise it becomes a Clusterhead.

Procedure *Link_failure(u);*

```
    begin
        if Ch(v) and (u ∈ Cluster(v))
           then Cluster(v):=Cluster(v)\{u}
            else if Clusterhead=u   then
                    if { z ∈ Γ(v): w_z > w_v ∧ Ch(v)}≠ Ø
                    then begin
                    x:=max w_z > w_v{z:Ch(z)};
                    send JOIN(v,x);
                    Clusterhead:=x;
                    end
           else begin
                send CH(v);
                Ch(v):=true;
                Clusterhead:=v;
                Cluster(v):={v};
            end
end;
```

- *New_Link.*When node v is made aware of the presence of a new neighbor u, it checks if u is a clusterhead. If this is the case,and if w_u is bigger than the weight of v's current clusterhead, than, independently of its own role, v affiliates with u.

Procedure New_Link(u);

```
begin
if Ch(u) then
      if (w_u> w_clusterhead)
      then begin
        send JOIN(v,u);
       Clusterhead:=u;
       if Ch(v) then Ch(v):=false
       end
end;
```

- On receiving CH(u).When a neighbor u becomes a clusterhead, on receiving the corresponding CH message, node v checks if its has to affiliates with u,i.e,it checks if whether w_u is bigger than the weight of v's clusterhead or not. In this case, independently of its current role, v joins u's cluster.

On receiving CH(u);

```
begin
    if(w_u > w_clusterhead) then begin
                    send Join(v,u);
                    Clusterhead:=u;
                    if Ch(v) then Ch(v):=false
                    end
end;
```

- On receiving JOIN(v,u).On receiving the message JOIN(u,z),the behavior of node v depends on whether it is clusterhead or not. In the affirmative, v has to check if either u is joining its cluster(z=v:in this case u is added to Cluster(v)) or if u is belonged to its cluster and is now is joining another cluster(z≠v:in this case, un is removed from Cluster(v)).If v is not a clusterhead, it has to check if u was its clusterhead. Only if this is the case, v has to decide its role:It will join the biggest clusterhead x in its neighborhood such that $w_x > w_v$ if such a node exists. Otherwise, it will be a clusterhead.

On receiving JOIN(u,z);

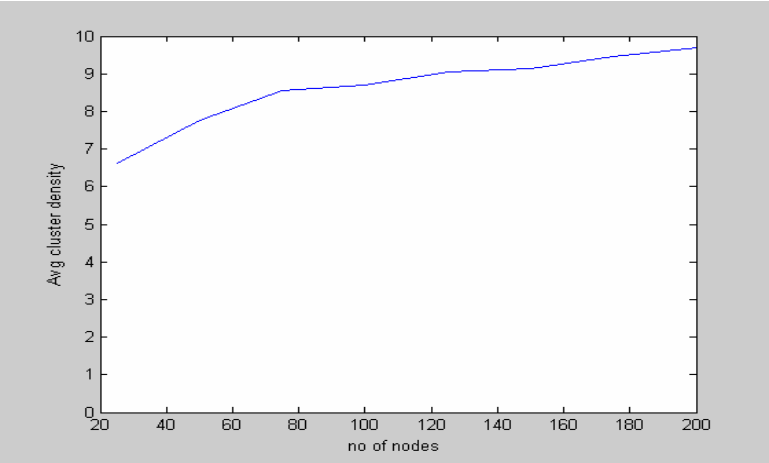
```
begin
  if Ch(v)
   then if z=v then Clusterhead(v):=Cluster(v)U {u}
        else if u Є Cluster(v) then Cluster(v):=Cluster(v)\{u}
    else if Clusterhead= u then
    if { z Є Γ(v): w_z > w_v Λ Ch(v)}≠ Ø
        then begin
                x:=max w_z > w_v{z:Ch(z)}:
                send JOIN(v,x);
                Clusterhead:=x;
               end
          else begin
                 send CH(v)
                 Ch(v):=true;
                 Clusterhead:=v;
                 Cluster(v):={v}
                end
end;
```

4.2 Simulation Results:

We simulated DMAC with no of nodes (N) € (10,200), Transmission range =10, Maximum displacement (mobility) =2 and in a 100X100 grid. We assumed δ (the maximum no of nodes a clusterhead can ideally support) =10, max disp=10, tRange € [0-60].Each node was assigned a random weight between 0 to 80.The following results were obtained:

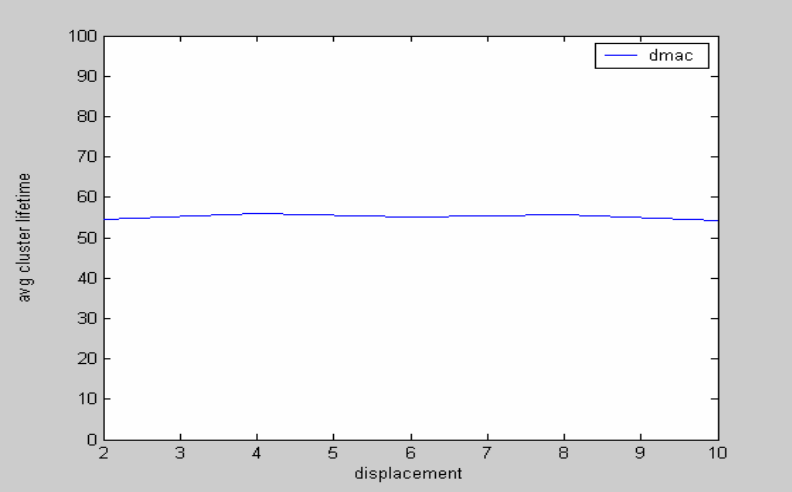
Average Cluster Density Vs no of nodes

As the no of nodes increase the average cluster density i.e. the average no of clusters tend to increase but the increase is not that prominent.



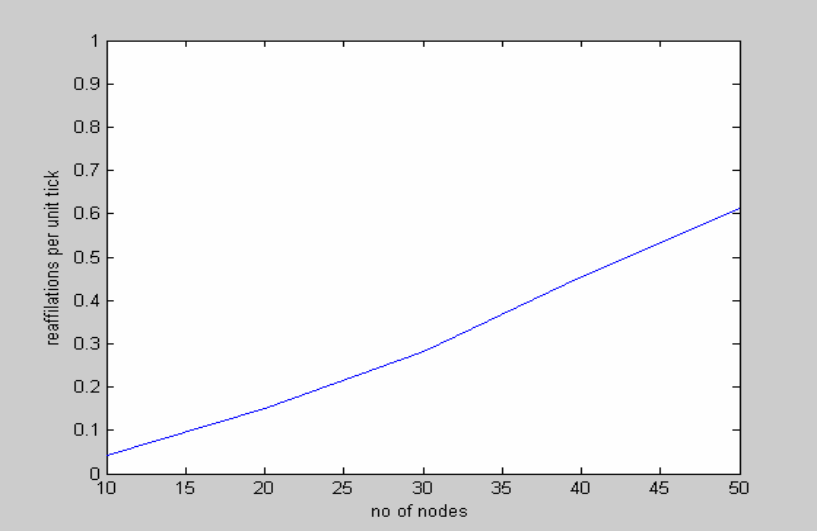
Average Cluster Lifetime Vs Maximum Displacement

Here the cluster lifetime remains constant because of the "Central Effect"[7] i.e. the nodes tend to cluster at the centre of the simulation area hence the maximum displacement has no effect on it.



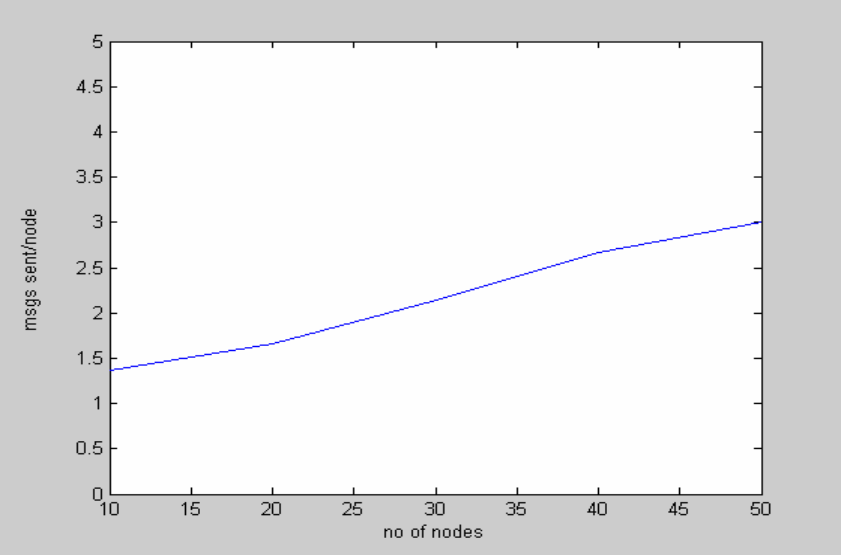
No of Reaffilations Vs no of nodes

As the no of nodes increase reaffilation per tick [2, 4] increase tremendously because the movement of the nodes become more erratic and they leave their clusters very frequently to join other clusters.



Message Complexity Vs no of nodes

Message complexity [2] is represented by the total no of messages sent by a node during cluster formation. As the no of nodes increase the nodes have to send more no of messages.



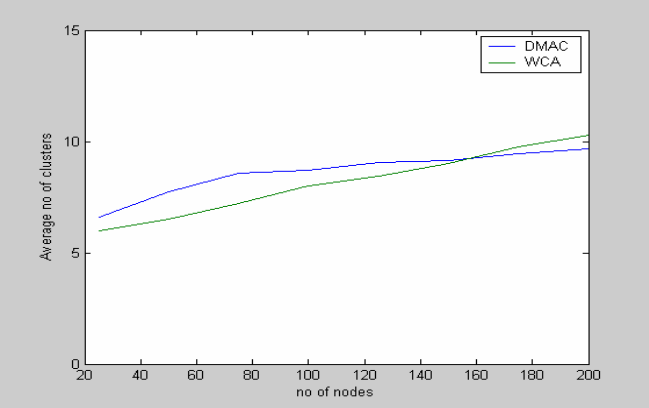
CHAPTER 5

Comparison between WCA and DMAC

Under the similar set simulating parameters (no of nodes (N) € (10,200), Transmission range =10, Maximum displacement (mobility) =2 and in a 100X100 grid. δ (the maximum no of nodes a clusterhead can ideally support) was assumed to be 10, max displacement=10, tRange € [0-60]) both the algorithms (WCA and DMAC) were compared taking the following set of parameters into account:

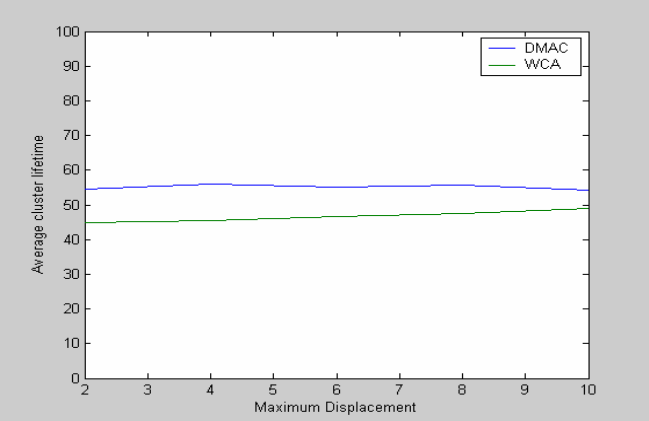
Average Cluster Density Vs no of nodes :

In the following graph it was observed that for WCA Average no. of clusters varied linearly with no of nodes, but in case of DMAC, it was almost constant as the no of nodes were increased, this is because the clusterhead election procedure in WCA is invoked very rarely while in DMAC it is done frequently.



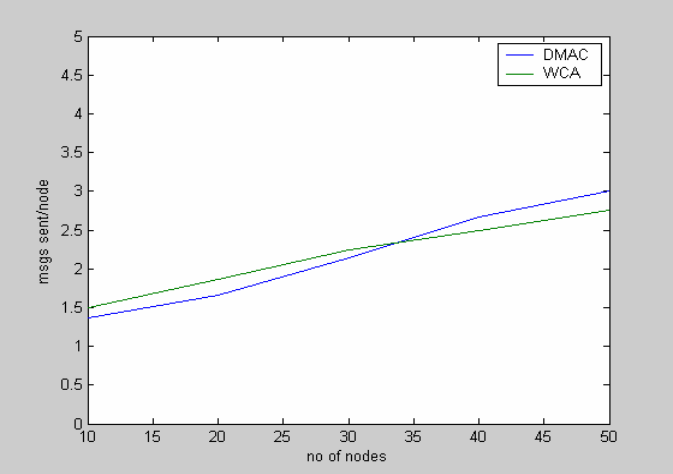
Average Cluster lifetime Vs maximum displacement:

The behavior of both the algorithms (WCA and DMAC) was almost similar while the above parameters were compared. This is due to the "Central effect"[7] i.e. the nodes tend to cluster at the centre of the simulation area.



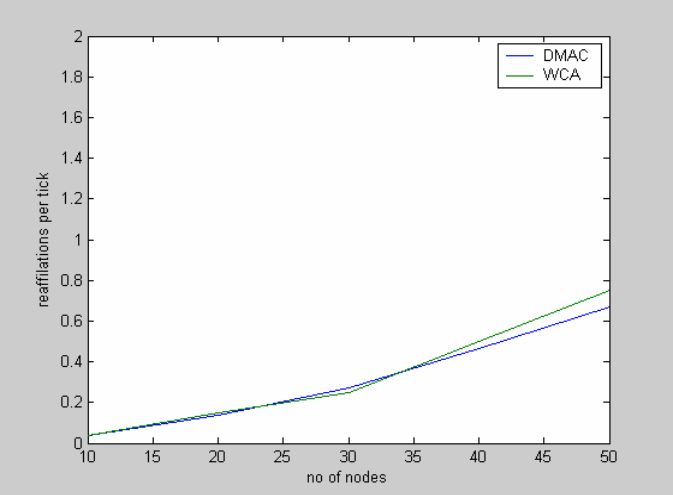
Messages sent/node Vs no of nodes:

For the lesser no. of nodes Average no. of messages sent by a node is greater for WCA but as the no. of nodes increase Average no. of messages sent becomes greater for DMAC .



Reaffilations per unit tick Vs no of nodes:

No. of reaffilations per tick is almost same for both the algorithms when the no. of nodes is low, but as the no of nodes increase the reaffilation rate is greater for DMAC.



CHAPTER 6

Conclusion

In this project we implemented both WCA and DMAC, recognized certain parameters to analyze the network performance, and compared both the models. It was observed that in certain cases WCA out performs DMAC while in some, the reverse is true. This is because in WCA, we give the weight of nodes the highest priority while in DMAC, we select the weights randomly. On the other hand we do not consider the mobility of the nodes during cluster formation in WCA but take it into account in case of DMAC. Hence DMAC can be merged with WCA to form a better clustering algorithm. This algorithm will capitalize the strengths of both the algorithms.

For a given no. of nodes whether the no. of clusterheads should be large or small is still an optimizing problem, because if the no. of clusterheads is large then to communicate with a given node, several nodes need to be traversed. Similarly if the no of nodes is small then the load on a clusterhead increases. Hence *Genetic Algorithms* can be used to optimize the no of clusterheads.

CHAPTER 7

REFERENCES

References:

1. BASAGNI,S. "Distributed Clustering For Ad Hoc Networks*", In proceedings of the 1999 International Symposium on Parallel Architecture, Algorithms and Networks ,Fremantle, Australia,* ,June 23-25,1999, pp. 1-16.

2. GHOSH, R., AND BASAGNI, S. "Limiting the Impact of Mobility on Ad Hoc Clustering". *PE-WASUN'05, October 10-13, 2005, Montreal, Canada*, pp. 197-204.

3. BASAGNI,S., CHLAMTAC,I., AND FARAGO,A . "A Generalized Clustering Algorithm for Peer-to-Peer Networks", *Proceedings of Workshop on algorithm aspects of communication (Satellite workshop of ICALP), Bologna, Italy July* 1997.

4. CHATTERJEE, M., DAS, S. K. AND TURGUT, D. , "WCA A Weighted Clustering Algorithm for Mobile Ad hoc Networks*", Journal of Clustering Computing, (Special Issue on Mobile Ad hoc Networks), Vol. 5, No. 2, pp. 193-204, April 2002.*

5. RAMACHANDRAN, L. , KAPOOR, M. , SARKAR, A. AND AGGARWAL, A., "Clustering algorithms for wireless ad hoc networks*", Proceedings of Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Boston, USA. pp. 54-63, 2000.

6. CHATTERJEE, M., DAS, S. K. AND TURGUT, D., "An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks", *Proceedings of IEEE GLOBECOM 2000*, San Francisco, November 2000, pp. 1697-1701

7. JARDOSH, A.,ELIZABETH M., ALMEROTH K. C. AND SURI,S., "Towards realistic mobility models for mobile ad hoc networks," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. 2003, pp. 217–229, ACM Press.

8. CAMP, T. , BOLENG, J. AND V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.