PROJECT REPORT ON

**“Project Title”**

Submitted By:

XXXX. Roll : 16901052013

XXXXX. Roll: 16901052012

1. Roll: 16901051011

# Under The Guidance Of:

Prof. XXX Department of XXXXX

# November, 2011



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ACADEMY OF TECHNOLOGY AEDCONAGAR, ADISAPTAGRAM HOOGHLY-712121, WEST BENGAL, INDIA**

# Certificate

This is to certify that XXX (Roll No. 16901052013), XXXXX (Roll No. 16901052012) and Sri. XXXX (Roll No. 16901051011) have successfully completed the project titled **“XXXX”** at Academy Of Technology under my supervision and guidance in the fulfillment of requirements of Seventh Semester, **Bachelor of Technology (Computer Science & Engineering)** of West Bengal University of Technology, Kolkata.

Prof. Prasenjit Das Prof. XXX (Project Guide )

Head of the Department Department of Computer Science and Engineering XXXXXXX

# Acknowledgement

We deem it a pleasure to acknowledge our sense of gratitude to our project guide Prof. XXXX under whom we have carried out the project work. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish to reciprocate in full measure the kindness shown by Mr. Prasenjit Das (H.O.D, Computer Science and Engineering) who inspired us with his valuable suggestions in successfully completing the project work.

We shall remain grateful to Mr. Prof. (Dr.) Dilip Bhattacharya, Director, Academy Of Technology, for providing us a strong academic atmosphere by enforcing strict discipline to do the project work with utmost concentration and dedication.

Finally, we must say that no height is ever achieved without some sacrifices made at some end and it is here where we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period of time.

Date: 08.05.2011

Place: AOT, AEDCONAGAR,

Hooghly, West Bengal

XXNameXXX Roll:

XXNameXXX Roll:

XXNameXXX Roll:

# Abstract

An ad-hoc network can be termed as a communication network for data transfer which does not have a permanent base station or a server. The Ad- hoc network is automated by a procedure where dynamically some network heads are selected which services the devices under it. These heads are selected based on the utilization and throughput at a certain region. The Mobile Ad-hoc network is one where the devices are mobile and in constant motion. So a particular server can never cater to all the devices. The mobile devices can only function properly and a uniformity in throughput can exist when the base stations are also mobile. Hence, a fixed topology in the arrangement of the devices can also never exist. So, in order to achieve the required performance, we have to create a dynamic network with a dynamic topology and also a dynamic base station. A normal device in the network only will be selected as the head and a fixed number of devices will be allocated to it in order to create this dynamic architecture. This device is the “Cluster Head”.

Our main objective is to develop a suitable protocol stack analogous to the

current TCP/IP stack which will perform according to the movement of the devices. The several parameters needed to be taken into account are:

* The number of devices.
* The pattern of movement of the devices and their speed.
* The distance between the devices in the co-ordinate space.
* The cluster head.

A Predictive Study will be done on the selection of cluster heads. The several parameters stated above will be varied and the newer values noted to develop the architecture of optimal performance. A statistical analysis at the end will prove the performance betterment and a better utilization of the Ad- hoc network.

# CONTENTS

|  |  |  |
| --- | --- | --- |
| **No.** | **Topics** | **Page** |
| 1. | Introduction | 6 |
| 2. | Software Requirements And Specifications | 13 |
| 3. | Project Plan | 15 |
| 4. | Design Strategy | 34 |
| 7. | Future Work | 77 |
| 8. | Conclusion | 79 |
| 9. | References | 80 |

* 1. **INTRODUCTON**
  2. **SOFTWARE REQUIREMENTS AND SPECIFICATIONS**

## Product Perspective

The purpose of this project work is to provide a proper clustering methodology and an efficient algorithm for the purpose of same. In this world of evolving mobile and wireless communications it is becoming increasingly important for us to understand and implement in the correct fashion the proper software for the communication purpose. A need for the increase in a low cost and low overhead system is very much in demand for the reduction of the net customer costs and survival in the market of intense competitions. So we aim at developing an algorithm which will prove to be more and more fruitful for this purpose and will be of immense applicability in the field of Mobile Ad Hoc Networks and Wireless Communications. The employment of newer concepts and algorithms will prove to be a much better headway in this field.

## User Characteristics

The users who are developing this whole application or are trying to make an evaluation of the proper working of the newest ideas and the algorithms implemented need to be proficient in the following areas :

* + - C++ programming including all the concepts of Object Oriented Programming and especially abstraction methods.
    - NED(Network Editor) language which is the language mainly employed for the purpose of development of the node structures and the network models.
    - OMNeT++ which is a discrete simulation software used for the purpose of development and testing of the whole system.

## Specific Requirements

The whole of this project work will be developed using the OMNeT++ simulator which uses the NED language for the purpose of the development of the network and the C++ language for the module definitions. So the specific requirements for this project work are :

* OMNeT++ version 3.0
* Microsoft Visual Studio.
* Windows XP or any compatible Operating Systems.

As an end product the algorithm developed by us will be implemented in mobile or other embedded devices operating under a wireless and mobile atmosphere. In that scenario the whole internal working will be abstracted from the actual users of the product and it is of no importance for them to understand the specific requirements of the programs executing inside their devices.

## Performance Requirements

The algorithms developed are purely based on a wireless domain and will be implemented on embedded devices as an end product. So it is not our concern about the performance of the embedded devices which is rather a concern of the company who will be implementing this technology in their products. For the purpose of development the only performance requirements are related to the specific software requirements mentioned above and a system which is capable of smoothly running these above said softwares is good enough for the project members to work on the proposed algorithms.

## Design Constraints

The OMNeT++ simulator is used as a visual aid only and can never bring in front of our eyes the actual scenario where millions of nodes exist in a single Ad Hoc Network. So, we must have this thing in mind that there exists a close approximation between the actual simulations developed and the practical measurements.

# 1.PROJECT PLAN

The major aspect of the project work has been to develop a proper clustering method and an efficient algorithm for the purpose of the same. The clustering process is done so as to make a better utilization of the resources available and as a good communication technique. The clustering process is carried out following the solution strategies stated before.

A detailed study of Mobility Models for Ad Hoc networks[4] was conducted as an initial step.

The study of Time Series Models for internet traffic [5],[6] have also helped in our development work.

The major plans that will be utilized for the purpose of clustering are:

* A Distributed Clustering Algorithm (DCA)[1].
* A Distributed Mobility Adaptive Clustering Algorithm (DMAC)[1].
* A Weight Based clustering algorithm (WCA)[2].
* A Voting Based clustering Algorithm (VCA)[3].

## Distributed clustering Algorithm(DCA) :

This algorithm is mainly developed to partition a given set of nodes into certain groups called clusters on the basis of weights which can be mainly applied on the field of quasi-static networks and or a fully mobile ad hoc network.

In an ad hoc network all the nodes are possibly mobile and it is of utmost importance to partition the set of nodes into groups, to make proper spatial reuse and use of the shared channel(i.e. in terms of time division of frequency division schemes) to minimize the amount of data to be exchanged to maintain the routing and control information in mobile

environment as well as building and maintaining cluster based virtual network architectures.

The first phase i.e. the *set-up phase* is accomplished by choosing some nodes that act as coordinators of the clustering process (clusterheads). Then the cluster is formed by associating a clusterhead with some of its neighbours(i.e. nodes within the head’s transmission range) that become the ordinary nodes of the cluster. Once the cluster is formed, the clusterhead can continue to be the coordinator of the operations in its cluster. A common assumption for the purpose of this clustering process is that the nodes do not move while the cluster formation is in progress. The biggest drawback in this assumption is that in a real ad hoc network this sort of an assumption can never be made as the nodes are always mobile in an ad hoc network.

Once the nodes are partitioned into clusters the non-mobility assumption is released and techniques are described on how to maintain the cluster formation in the presence of mobility. Often a reorganization of clusters is done which occurs due to node mobility by invoking the clustering process again. The re clustering process cannot rely on cluster organization. Thus this is a feasible solution strategy only when the network does not need too many re organizations. So for the discussion of the above said method the need arises for the ordinary node to have :

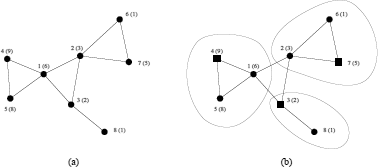
* + 1. At least one neighbouring clusterhead(which allows fast communication among any pair of nodes).
    2. To be affiliated with the best neighbouring cluster head.
    3. Need to have a clusterhead that is well scattered throughout the network.

**Preliminaries and Problem Definition:**

We model an ad hoc network by an undirected graph G=(V,E), in which V,

|V|=n, is a set of wireless nodes and there is an edge {u,v}E if and only if u and v can mutually receive each other’s transmission. In that case u and v are neighbours. The set of neighbours of a node vV will be denoted

by (v). Due to mobility the graph can change in time. Every node is assigned an unique identifier(ID). For simplicity here we identify each node with its ID and we denote both with v. Finally, we consider the weighted network i.e., a weight *wv* (a real number>0) is assigned to each node vV of the network.



**Fig 3.1 (a)** : The Network before the clustering is initiated

**Fig 3.1(b)** : The Network after the clustering is complete.

In order to meet the requirements imposed by the wireless mobile nature of the networks, a clustering algorithm is required to partition the nodes of the network so that the following ad hoc clustering properties are satisfied:

1. Every ordinary node has at least a clusterhead as its neighbour(*dominance property*).
2. Every ordinary node affiliates with the neighbouring clusterhead that has the bigger weight.
3. No two clusterhead can be neighbours(*independence property*).

The first property is necessary to ensure that each ordinary node has direct access to at least one of the cluster heads(of the cluster to which it belongs), thus allowing fast intra and inter-cluster communications.

The DCA[1] algorithm is implemented at each node in such a way that a node v decides its own role(clusterhead or ordinary node) depending solely on the decision of its neighbours with bigger weights. Thus, initially

only those nodes with the bigger weight in their neighbourhood will broadcast messages to their neighbours stating that they will be clsuterheads. On receiving one or more of this “clusterhead” message a node v will decide to join the cluster of that neighbouring clusterhead with the biggest weight. If no node with bigger weight have sent such a message, then v will send a clusterhead message. We will find that all the nodes terminate the algorithms being either clusterheads or ordinary nodes, and that the ad hoc clustering properties are satisfied. The entire algorithm is message driven and a specific procedure will be implemented corresponding to a particular message. We use two types of messages : CH(u), used by a node v to make neighbours aware that it is going to be a clusterhead and JOIN(u,v) with which a node v communicates to its neighbours that it will be part of the cluster whose clusterhead is node u(v). In the procedures, we use these notations:

* *v*, the generic node executing the algorithm.
* *Cluster(v):* the set of nodes in v’s cluster. Initialized to null.
* *Clusterhead:* The variable in which every node records the ID of the clusterhead it joins.
* *Ch(-) and Join(-,-)* are Boolean variables. Ch(u) is set to true if it sends or receives a CH(v) message. Join(u,t) is set to true by v when it receives JOIN(u,t) message from u.
* By executing EXIT the node quits execution of the algorithm.

## The init procedure:

Every node starts its execution at the same time by executing the procedure Init.

PROCEDURE INIT;

#### begin

**if**

**then begin**

**send** *CH(v);*

*Cluster(v) := Cluster(v)*  {v};

*Ch(v) :=* **true;**

*Clusterhead :*= v

#### end

**end**

**On Receiving Ch(u) :**

On receiving a Ch(u) message from a neighbour u, node v checks if it has received all its neighbours z such that wz > wu, a JOIN(z,x) message, xV( recorded in corresponding Boolean variable). In that case ,v will not receive a CH message from these z, and u is the node with the biggest weight in v’s neighbourhood that sent a CH message.

ON RECEIVING CH(u);

#### begin

*Ch(u)=***true**;

#### if then begin

*Clusterhead* := u;

**send** JOIN(v,*Clusterhead*); EXIT

#### end

**end**

**On Receiving Join(u,t) :**

On receiving a Join(u,t) message, node v checks if it has previously sent a CH message(i.e., it has already decided to become a clusterhead). In that case, it checks if node u wants to join v’s cluster(v=t), and possibly

updates the Cluster(v) set. Then if, all the v’s neighbours z such that wz<wv have communicated their willingness to join a cluster, v quits the DCA execution. If node v has not sent a CH message before, it needs to know what all nodes z such that wz>wv have decided for themselves. If v has received a message from all such nodes, then it checks the nature of the messages received. If they are all JOIN messages, this means that all those neighbours z have decided to join a cluster as ordinary nodes. This implies that now v is the node with the biggest weight among the nodes that have still to decide what to do. In this case, v will be a clusterhead and it executes the needed operations. At this point v also checks if each neighbour y such that wy<wv has already joined another cluster. If this is the case, v quits the algorithm execution: it will be the clusterhead of a cluster with a single node.

ON RECEIVING JOIN(u,t);

#### begin

*Join(u,t)* := **true**; **if** *Ch(v)*

#### then begin

**end else if**

**if** v=t **then** *Cluster(v) := Cluster(v)*  {u};

**if ** **then** EXIT



#### then if

**then begin**

**send** *CH(v);*

*Cluster(v) := Cluster(v)*  {v};

*Ch(v)=***true**; *Clusterhead :*= v;



**if then** EXIT

#### end else begin

*Clusterhead :*= maxwz{z: Ch(z)}; **send** JOIN(c,*Clusterhead*); EXIT

#### end

**end**

Hence this above DCA algorithm is implemented in our simulation and the various parameters are studied.

## Distributed Mobility Adaptive Clustering (DMAC) [1]:

Here we describe a distributed algorithm for the set up and the maintenance of a cluster organization in the presence of node’s mobility that satisfies the three ad hoc clustering properties. The main difference with the DCA presented in the previous section consists in the fact that here we do not assume that during the clustering process the nodes of the network need not to move. This makes the algorithm suitable for both the clustering set up and the maintenance, which was not possible in the previous solutions. Adaptation to changes in the network is 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 DMAC[1] we will assume that a message sent by a node is received correctly within a finite time(a step) by all of its neighbours. We also assume that each node knows its own ID, its weight, its role and the ID, the weight and the role of all its neighbours. When a node has not yet decided what its role is going to be, it is considered as an ordinary node. Furthermore we assume that :

* Every node is made aware of the failure of a link, or of the presence of the new link by a service of the lower level.
* The procedures of DMAC[1] are “atomic” i.e., they are not interruptible.
* At clustering set up time or when a node is added to a network its variables *Clusterhead, Ch(-)* and *Cluster(-)*are initialized to nil or false.

## Init:

At the clustering set up, or when a node v is added to the network, it executes the procedure Init in order o determine its role. If among its neighbours there is at least a clusterhead with a bigger weight then v will join it. Otherwise it will be a clusterhead.

PROCEDUE INIT;

#### begin

**if** {z(v) : wz>wv  *Ch(z)* ≠ 

#### then begin

x:= maxwv {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 the case v removes u from the *Cluster(v).* If v is an ordinary node and u was its 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(v) such that wz>wv.

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  (v) : wz > wv  *Ch(z)*} ≠ 

#### then begin

x:=maxwx>wv {z : *Ch(z)*};

**send** JOIN(v,x);

#### end else begin

**send** *CH(v); Ch(v) :=* **true**; *Clusterhead :=* v; *Cluster(v) :=* {v}

#### end

**end**

**New Link :**

When a node v is made aware of the presence of a new neighbour u, it checks if u is a clusterhead. If this is the case and if wu is bigger than the weight of v’s current clusterhead, then independently of its own role, v affiliates with u.

PROCEDURE NEW\_LINK(u);

#### begin

**if** *Ch(u)* **then**

**if** (wu > wClusterhead)

#### then begin

**send** JOIN(u,v);

*Clusterhead* := u;

**if** *Ch(v)* **then** *Ch(v)* := **false**

#### end

**end**

**On Receiving Ch(u) :**

When a neighbour u becomes a clusterhead, on receiving the corresponding CH message, node v checks if it has to affiliate with u i.e., it checks whether wu is bigger than the weight of v’s clusterhead or not.

ON RECEIVING CH(u);

#### begin

**if** (wu > wclusterhead) **then begin**

#### end

**end**

**send** JOIN(v,u);

*Clusterhead :*= u;

**if** *Ch(v)* **then** *Ch(u)* := **false**

## On Receiving JOIN(u,z) :

On receiving the message Join(u,z), the behaviour of a node v depends on whether it is a clusterhead or not. In the affirmative, v has to check if either u is joining its cluster. If v is not a clusterhead, it has to check if u was a clusterhead. Only of this is the case, v has to decide its role. It will join the biggest node in the neighbourhood such that wx > wv if such a node exists.

ON RECEIVING JOIN(u,z);

#### begin

**if** *CH(v)*

**then if** z=v **then** *Cluster(v)* := *Cluster(v)*  {u}

**else if** u  *Cluster(v)* **then** *Cluster(v) := Cluster(v)* \ {u}

**else if** *Clusterhead*=u **then**

**if** { z  (v) : wz > wv  *Ch(z)*} ≠ 

#### then begin

**end else begin**

x := max wz>wv {Z : *Ch(z)*} **send** JOIN(v,x); *Clusterhead* := x;

#### end

**end**

**send** CH(v); *Ch(v)* := **true**; *Clusterhead* := v; *Cluster(v)* := {v}

Hence we conclude this section by showing that by using these procedures we can obtain any ad hoc network clustering, that always satisfies the ad hoc clustering properties.

## Weight Based clustering Algorithm(WCA) [2]:

A multi-cluster, multi-hop packet radio network architecture for wireless systems should be able to dynamically adapt itself with the changing network configurations. Certain nodes, known as *clusterheads*, are responsible for the formation of *clusters* each consisting of a number of nodes (analogous to *cells* in a cellular network) and maintenance of the topology of the network. The set of clusterheads is known as a *dominant set*. A clusterhead does the resource allocation to all the nodes belonging to its cluster. Due to the dynamic nature of the mobile nodes, their association and dissociation of nodes to and from *clusters* perturb the stability of the network topology, and hence a reconfiguration of the system is often unavoidable. However, it is vital to keep the topology stable as long as possible. The *clusterheads*, form a *dominant set* in the network, determine the topology and its stability. The weight-based distributed clustering algorithm takes into consideration the ideal degree, transmission power, mobility, and battery power of mobile nodes.

Here, we study a weight based distributed clustering algorithm[2] which takes into consideration the number of nodes a clusterhead can handle ideally (without any severe degradation in the performance), transmission power, mobility, and battery power of the nodes. Unlike other existing schemes which are invoked periodically resulting in high communication overhead, our algorithm is *adaptively* invoked based on the mobility of the nodes. The algorithm achieves *load balancing* by specifying a pre-defined threshold on the number of nodes that a clusterhead can handle ideally. This ensures that none of the clusterheads are overloaded at any instance of time. The *load balancing factor* (LBF) is defined to measure the degree of load balancing among the clusterheads. The term *connectivity* defines the probability that a node is reachable from any other node. Clusterheads in this scheme work in dual power mode. The clusterheads can operate at a higher power mode (resulting in a higher transmission range) for inter- cluster communication while they use lower power for intra-cluster communication. Finally, the detailed simulation experiments demonstrate that the clustering algorithm yields better results as compared to the existing heuristics in terms of the number of re affiliations and dominant set updates.

The network formed by the nodes and the links can be represented by an undirected graph

*G* = *(V ,E)*, where *V* represents the set of nodes *vi* and *E* represents the set of links *ei* . Note that the cardinality of *V* remains the same but the cardinality of *E* always changes with the creation and deletion of links. More formally, we look for the set of vertices *S V (G)*,such that



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



where *tx*range is the transmission range of *v*. The neighborhood of a cluster head is the set of nodes which lie within its transmission range. The set *S*

is called a *dominating set* such that every vertex of *G* belongs to *S* or has a neighbor in *S*.

## Basis of the algorithm :

To decide how well suited a node is for being a clusterhead, the algorithm take into account its degree, transmission power, mobility and battery power. The following features are considered in the clustering algorithm:

* The cluster head election procedure is not *periodic* and is invoked as rarely as possible. This reduces system updates and hence computation and communication costs. The clustering algorithm is not invoked if the relative distances between the nodes and their clusterheads do not change.
* Each clusterhead can ideally support only *δ* (a pre-defined threshold) nodes to ensure efficient 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 the nodes will incur more delay because they have to wait longer for their turn (as in TDMA) to get their share of the 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 close distance to each other. A clusterhead consumes more battery power than an ordinary node since a clusterhead has extra responsibilities to carry out for its members.
* *Mobility* is an important factor in deciding the clusterheads. In order to avoid frequent clusterhead changes, it is desirable to elect a clusterhead that does not move very quickly. When the clusterhead moves fast, the nodes may be detached from the clusterhead and as a result, a *re affiliation* occurs. Re affiliation takes place when one of the ordinary nodes moves out of a cluster and joins another existing cluster. In this case, the amount of information exchange between the node and the corresponding clusterhead is local and relatively small.

The information update in the event of a change in the dominant set is much more than a re affiliation.

* A clusterhead is able to communicate better with its neighbors having closer *distances* from it within the transmission range. As the nodes move away from the clusterhead, the communication may become difficult due mainly to signal attenuation with increasing distance.

## Clusterhead selection procedure :

**Step1.** Find the neighbors of each node *v* (i.e., nodes within its transmission range) which defines its *degree*, *dv*, as



**Step 2.** Compute the *degree-difference*, ∆*v* = |*dv* − *δ*|, for every node *v*.

**Step 3.** For every node, compute the *sum of the distances*, *Dv*, with all its neighbors, as



**Step 4.** Compute the running average of the speed for every node till current time *T* . This gives a measure of mobility and is denoted by *Mv*, as



where *(Xt, Yt )* and *(Xt*−1*, Yt*−1*)* are the coordinates of the node *v* at time *t* and *(t* − 1*)*, respectively.

**Step 5.** Compute the cumulative time, *Pv*, during which a node *v* acts as a clusterhead. *Pv* implies how much battery power has been consumed which is assumed more for a clusterhead than an ordinary node.

**Step 6.** Calculate the *combined weight Wv* for each node *v*, where



where *w*1*,w*2*,w*3 and *w*4 are the *weighing factors* for the corresponding system parameters.

**Step 7.** Choose that node with the smallest *Wv* as the clusterhead. All the neighbors of the chosen clusterhead are no longer allowed to participate in the election procedure.

**Step 8.** Repeat steps 2–7 for the remaining nodes not yet selected as a clusterhead or assigned to a cluster.

## Voting Based Clustering Algorithm(VCA)[3] :

Energy consumption is a critical issue in sensor networks. Even with the latest energy-efficient circuit design, limited battery power hinders the deployment of wireless sensor networks. To reduce energy consumption, energy-efficient data dissemination techniques are needed. One of the major data dissemination methods is external storage in which data need to be forwarded to a single sink outside the network. The sink location is often determined before all sensors are placed and a mobile sink is uncommon with current technology. Clustering is a useful approach to reduce energy dissipation in sensor networks. It is often coupled with data fusion to extend sensor lifetime. Each cluster elects one node as the cluster head. Data collected from sensors are sent to the cluster head first, and then forwarded to the sink. Compared to the sink, a cluster head is closer to sensors within the same cluster. Cluster heads can fuse data from sensors to minimize the amount of data to be sent to the sink. This can greatly reduce the energy consumption of sensor networks. When the network size increases, clusters can be organized hierarchically to further reduce energy consumption. **Voting-based clustering algorithm (VCA) [3]**for energy-efficient data dissemination in wireless sensor networks is capable of combining load balancing, topology and energy information together. Furthermore, VCA does not make any assumptions about sensor location and network topology. It is fully distributed and energy-efficient.

Simulation results show that the algorithm can generate a longer network lifetime than many traditional energy efficient clustering approaches.

## Problem Statement

Since a cluster head is responsible for fusing the data from sensors subscribing to it, only one data packet needs to be delivered to the sink out of a cluster. Thus the more clusters are present, the more messages need to be delivered to the sink. If the sink is far away from sensors, minimizing the number of clusters reduces overall energy consumption. However, finding the optimal cluster heads to cover a sensornet is NPcomplete. Load balancing is another vital metric for distributed clustering algorithms. Sensors with high residual energy should have a higher opportunity of becoming cluster heads. Additionally, more sensors should subscribe to cluster heads with higher residual energy. However, many clustering algorithms try to balance the size of the clusters instead of the energy distribution.

## Vote Calculation[3]

The heuristic behind sensor voting is that a sensor’s importance should be reflected from all its neighbors (including itself) instead of from its local properties alone. Each sensor calculates the suitability that one of its neighbors (including itself) becomes its cluster head and casts a vote on that neighbor. At the same time, sensors collect votes from their neighbors and calculate the total vote received. The more votes a sensor accumulates, the more important it is in the whole network. During the clustering phase, sensors compete with each other based on the total votes each has received. For a sensor vi, the vote it casts on another sensor vj is:

### (1)

 (2)

The total vote of sensor vi is the sum of the votes from all its neighbors.

**Observation 1:** If the network is fully connected (all sensors can communicate with each other), the sensor with the highest residual energy becomes the cluster head. When the network is fully connected, all

sensors cast the same vote of on sensor vi. As a result, the higher residual energy a sensor has the higher vote it receives. Since high energy nodes are more likely to become cluster heads, VCA can balance the energy distribution across a sensor network by consuming the power of high energy nodes first.

## Load Balancing

To balance the workload among cluster heads, the algorithm implemented two different strategies when a sensor is within the communication range of multiple cluster heads. The first strategy is for a node to join the cluster head with the minimum node degree. This strategy tries to balance the size of each cluster. The fitness of a sensor vi is defined as:



## VCA: Voting-based Clustering Algorithm

To calculate the vote of each neighbor, a sensor needs to know the residual energy of all its neighbors. It is assumed that sensor energy changes slowly over time. Since sensors are quasi-stationary, they need to send heartbeat messages regularly to update all their neighbors. Residual energy information can be attached to these heartbeat messages to reduce communication overhead. The algorithm assumes that all sensors are synchronized. After receiving a residual energy message, a sensor needs to update its vote to all neighbors. Only one message is needed to broadcast votes to all neighbors. Note that a small residual energy change of a neighbor does not change the votes significantly.

**Theorem1:** Assume sensors are dispersed in an area S=[0, L]2, the time complexity of VCA is

**Theorem 2:** The message complexity of VCA is O(N) **.**

# DESIGN STRATEGY

The main design procedure in this project work of ours has been carried out with the simulations being done in OMNeT++ which is a discrete simulation software developed to display and simulate communication networks. OMNeT++ is an open source simulation package. Development started in 1992, and since then different contributors have written several models for it. Some of these models simulate simple queuing models, others simulate more realistic protocols such as TCP/IP. OMNeT++ is used by universities and companies like Lucent and Siemens. Before the entire

working of the project work took its full shape we first learnt the whole working of the communication system and then we constructed the sample simulations which were then employed for the purpose of studying the Mobile Ad-Hoc Networks. The stress is laid more towards the development of a layered architecture analogous to the TCP/IP protocol suite that can be employed in a proper way to implement the working of the whole simulation. We have used the OMNeT++ software as it integrates the visual analysis along with the technical or the architectural aspects of the whole simulated system.

## Approach

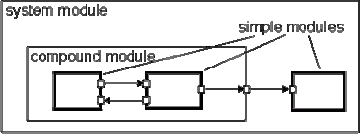
First, the OMNeT++ package was needed to be studied. After that, the simulations were developed as a slow and gradual process. The Mobile Ad Hoc model[4] was studied and the simulation for this was executed in OMNeT++. The outcomes from these tests have been compared with the published results and theoretical outcomes.

## OMNeT++ :

OMNeT++ is a discrete event simulator in development since 1992. Its primary use is to simulate communication networks and other distributed systems. It runs on both Unix and Windows. Other similar programs are the widely used NS-2 , which has extensive support for TCP, and OPNET.

OMNeT++ started with a programming assignment at the Technical University of Budapest (Hungary), to which two students applied. One of them, András Varga, still is the maintainer of this open source simulation package. During the years several people contributed to OMNeT++, among which were several students from the Technical University of Delft (The Netherlands) and Budapest. Milestones in the development are the first public release in 1997 and the added animation functionality in 1998, which made the package even more usable for education.

## NED :

A simulation in OMNeT++ is written in two different languages, NED and C++. These two languages separate the design of a topology and the implementation of the modules that exist in this topology. The NED (NEtwork Description) language is used to describe the layout (topology) of the simulation.

**Fig 4.1** : Visual Network Module Layout

As we we can see in Figure, each simple module described by the NED language can also be used in a large compound module, which is also described in the NED language. One simple module can for example be a TCP implementation, the other one an Ethernet implementation, and together they can form an Internet host (the compound module). There are three distinctive elements that can be described in a NED file:

* **Module definitions**. In these blocks we can describe simple and compound modules and set parameters of these modules. Also, we can define gates that connect the modules.
* **Channel definitions**. Describe channels (links) that connect modules.
* **Network definitions**. To get the whole simulation running, we’ll have to describe which module is the top level module for the network we wish to simulate. This top level module is an instance of the system module.

## Module implementation in C++ :

The implementation of the modules defined in the NED language is done in C++. We can use the OMNeT++ simulation library, with which we can simulate any kind of behaviour in our models. This library class includes mechanisms to schedule and cancel events, gather statistical data and produce data that can be inspected (plotted) using the graphical tool ‘Plove’, which is also included in the package.

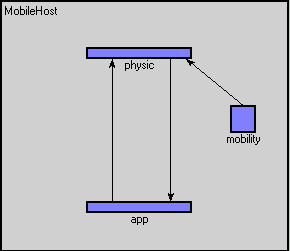
## Running a simulation :

A simulation in OMNeT++ can be run in two different ways: visual and text-only. The visual simulations are shown in the following graphics. This way of running the simulation is particularly useful when first running the simulation, or to get acquainted with the protocols or networks the program simulates. It shows all the messages that are exchanged between the modules in an animation. Also, with larger simulations, we can look deeper into each module, to see what messages are exchanged internally.

## Node Architecture:

The nodes that have been constructed for our project work have been made by taking three layers inside each node each with its own specific purpose. The three layers work analogous to the TCP/IP suite ad each layer provides services to its adjacent layers. The three layers are :

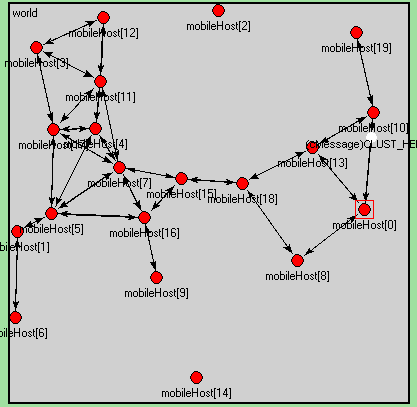
* **The Physic layer** : It is similar to the physical layer whose main job is to transfer the raw bits of data across the network to the other nodes and to the adjacent devices and also to receive bits from them. The Physic layer obtains the bits from the Application layer lying above to it which supplies it with the control informations as well as the pay-load. Similarly the bits received from the other neighbouring nodes by the physic layer are transmitted upwards to the Application layer by the physic layer. The other important function of the Physic layer is to dynamically construct the topologies of the wireless network in the presence of mobility.
* **The Application Layer** : The Application layer is like the brain of the node where all the processing of the informations take place. The clustering procedures are entirely carried out here at the Application layer and the algorithms developed are being executed in the modules of the Application layer. The Application layer is entirely responsible for the selection of the clusterheads and the clusters to which a particular node will be joining.
* **The Mobility Layer** : The mobility layer is entirely responsible for lending the mobile nature to the ad hoc network and it does so by providing a new location for the nodes at regular intervals of time. The Mobility layer also initiates the execution of the clustering process by sending the initial messages.



**Fig 4.2**: The Node Architecture

## Network Architecture:

The network architecture is formed arbitrarily. As the field of our study is the mobile ad hoc networks so we consider the placement of a random network architecture where all the nodes are spread out evenly all over the coordinate space. The space is prefixed before the start of the simulation and the nodes are distributed in an as uniform manner as possible.



**Fig 4.3:** The Network Architecture.

# FUTURE WORK

The future work of this project is mainly based on the development of the following features which can be easily incorporated in the existing algorithm that has been proposed by us.

* **Predictability of nodes**: This is the area which can be highly worked upon in the future by developing the ability to guess the number of nodes that are going to exist in the near future. That is we have to develop a

model based on the statistical studies where we will predict the number of nodes that will come under a particular cluster in the near future based on the values of some past clusters. This will enable the cluster nodes to make advance membership to certain cluster heads and will not require undergoing the whole clustering process all over again due to which the net performance of the algorithm will further increase.

* **Fuzzyfication**: The concept of fuzzy clustering also can be implemented into this project work where we may move away from the concept of crisp membership of a node in a cluster to some partial membership of a node in one or more clusters, so that, a node can be a member of more than one clusters, which is a more realistic phenomenon. We now discuss the properties of clustering which we have already discussed in the section 3.2:
  + Dominance Property: This initially stated that the node cannot be a member of more than one clusters at a particular time. From the fuzzy point of view this concept can be changed and fuzzified to say that a node can be a member of more than one particular cluster head based on the distance from the different cluster heads; i.e., depending upon the distance from each, it may have a varying degree of belonging to different clusters.
  + Only the node with highest weight can be a cluster head: This concept can be re worked and stated as that the cluster with the maximum number of nodes physically closer to it has the capacity to become the cluster head and not the one with the highest weight. In this way there could be more than one nodes which can be a part of two or more clusters with varying membership values.
  + Independence Property: This property states that no two cluster-heads can be neighbours. So, the property of fuzzzyfication can be stated as that two cluster-heads could be neighbours depending upon the membership values which are determined by the number of nodes under its cluster set. This property will be a huge difference to the

existing concepts and the whole clustering process can be made to be more flexible.

These above two concepts is on the process of being implemented and the results obtained so far show further development of the existing performance parameters. The fuzzyfication of the whole network is a huge boost to the overall performance parameter as it makes the net performance much higher.

# CONCLUSION

The work of Clustering in Mobile Ad Hoc Network is a research based project work which keeps on taking its own course along the whole timeline of work. We are having fixed in our minds certain targets which we have enumerated before in this report. So, we find it important to take stock of the whole work that we have done till now. The understanding of

the proper functioning of the Mobile Ad Hoc Network is totally based upon thorough theoretical study. The simulations and the practical implementation of the new ideas have been successfully developed by means of the network architecture and coding of the various modules of the working model. We also have made a considerable progress regarding the forecasting capability of the algorithm.

The detailed study of the program codes reveal that the entire model is devoid of any complicated nested loops and is based on simple code fragments, leading to its average complexity being O(n). Implementation of the prediction algorithm will make the model much more effective with respect to time , cost and space.

In our future work we plan to implement the following:

* The Node Prediction System where we will be implementing in our algorithm, the code for the Time Series analysis , to predict the future location of a particular node.
* Fuzzyfication: We plan to introduce the Fuzzy Set Theory into our algorithm to find out the membership of a node in a cluster. We shall try to find out the degree of participation of a node for a clusterhead on the basis of its membership values which will be dictated by the Fuzzy Theory.

# REFERENCES

1. : Stefano Basagni., “Distributed Clustering for Ad Hoc Networks”, in Proc.

*International Symposium on Parallel Architectures, Algorithms and Networks*, June 1999, pp. 310-315.

1. : Mainak Chatterjee, Sajal Das and Damla Turgut, “WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks”, *Journal of Cluster Computing, Special issue on Mobile Ad hoc Networking*, No. 5, 2002, pp. 193-204
2. : Min Quin, Roger Zimmermann ,“VCA: An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks”, *Journal of Universal Computer Science, vol. 13,* no. 1 (2007),pp. 87-109
3. : T. Camp, J. Boleng, V. Davies , “A Survey of Mobility Models for Ad hoc Network, Special issue on Mobile Ad hoc Networking: Research, Trends and Applications”, *vol. 2*, no. 5, pp. 483-502, 2002.
4. : Y. Borgne, S. Santini, G. Bontempi, “Adaptive Model Selection for Time Series Prediction in Wireless Sensor Networks”, *International Journal for Signal Processing” , vol. 87,* no. 12, pp 3010-3020,2007
5. : C you, K. Chandra, “ Time series models for internet data traffic”, *In Proceedings of the 24th. Conference on Local Computer Networks LCN 99,* pp 164-171, 1999.