TRAFFIC OPTIMIZATION AND MINIMIZATION OF PACKET LOSS THROUGH K-MEANS CLUSTERING IN WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

ABHISHEKK CHANDU	71203104001
C.V.HARI KRISHNAN	71203104011
JITAIN K. RAHEJA	71203104015

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641006

ANNA UNIVERSITY: CHENNAI 600 025

April 2007

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Traffic Optimization and Minimization of Packet loss through K-means clustering in Wireless Sensor Networks**" is the bonafide work of

ABHISHEKK CHANDU	**71203104001**
C.V.HARI KRISHNAN	**71203104011**
JITAIN K. RAHEJA	**71203104015**

who carried out the project work under my supervision.

SIGNATURE
Dr.S.Thangaswamy,Ph.D.
Dean/HEAD OF DEPARTMENT,
Department of Computer Scince & Engg,
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore-641006.

SIGNATURE
Ms.Amutha Venkatesh M.E.
SUPERVISOR,
Senior Lecturer,
Department of Computer Scince & Engg,
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore-641006.

Submitted for viva-voice examination held on 25th APRIL 2007 .

Internal Examiner 25.4.07

External Examiner 25/4/07

DECLARATION

We hereby declare that the project entitled" **TRAFFIC OPTIMIZATION AND MINIMIZATION OF PACKET LOSS THROUGH K-MEANS CLUSTERING IN WIRELESS SENSOR NETWORKS**" is a record of original work done by us and to the best our knowledge, a similar work has not been submitted to Anna University or any institutions, for fulfillment of course study.

The report is in partial fulfillment of the requirements for the award of the degree of Bachelor of Computer Science and Engineering of Anna University. Chennai.

Place: Coimbatore
Date: 25/04/07

Abhishekk Chandu

(Abhishekk Chandu)

(C.V.Hari Krishnan)

Jitain K. Raheja

(Jitain K. Raheja)

ACKNOWLEDGEMENT

The exhilaration achieved on successful completion of any task should be shared with the people behind the venture. At the onset, we thank the management of our college for having provided the excellent facilities to carry out the project.

We express our deep gratitude to our principal Dr. Joseph V. Thanickal for ushering us in the path of triumph.

We are always thankful to our beloved Professor, Dean and HEAD of Computer Science and Engineering Department, Dr.S.Thangasamy, whose consistent support and enthusiastic involvement helped us a great deal.

We convey our sincere thanks to our project coordinator assistant Professor Ms.Rajini, for her invaluable assistance.

We are greatly indebted to our beloved guide Ms. Amutha Venkatesh, Senior Lecturer, Department of Computer Science and Engineering for her excellent guidance and timely support during the course of this project.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We humbly acknowledge the inspiration received from the Almighty without whose help the project would have not taken shape.

ABSTRACT

Wireless Sensor Network is one promising application on wireless ad hoc networks. Sensor Networks can monitor ambient conditions such as temperature, sound, light and others. Information is collected from many sensor devices for further consumer application in the Sensor Network. The hierarchical management architecture can be applied to handle numerous sensor nodes. The lower-level nodes are managed and organized by the higher-level nodes by using 20/80 rules. Controlling the top-level nodes can decrease the costs of managing nodes and the communication among them. The 20/80 rule is a well-known "leadership shorthand term", originates from Vilfredo Pareto. who discovered a common phenomenon: about 80% of the wealth in most countries was controlled by a consistent minority around 20% of the people in most countries.

This project proposes a scheme for Self- Organization Management Protocols of higher-level nodes to contest member nodes with multi-hop form clusters, introduce the "20/80 Rule" for determining the ratio of headers to member nodes. This study implements the proposed management protocols including Clustering Mechanism for constructing cluster headers to solve the problems of clustering and broadcast storm, the suitable protocol to provide low cost communications between clusters.

First the Location information of the sensor nodes is to be collected from source and the cluster heads were elected/selected based on the locations of the active sensor nodes in network. For selecting a cluster head, k-means algorithm is used to find the cluster center. The nearby sensor nodes of a cluster-head then forward their data to sink only via the cluster-head. For hop to hop packet forwarding, AODV protocol is

used. That is to forward a packet from a sensor node to a cluster-head or cluster head to sink; normal routing protocols may be used.

The proposed clustering based mechanism reduces the number of management nodes in a large-scale sensor network,therby optimizing network traffic and minimization of packet loss. All the simulations of the proposed idea is simulated on ns2 network simulator and the performance of the proposed scheme is evaluated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Introduction

Wireless sensor networks represent a new data collection paradigm in which adaptability plays important role. Wireless sensor networks are comprised of a vast number of ultra-small fully autonomous computing, communication and sensing devices, with very restricted energy and computing capabilities that co-operate to accomplish a large sensing task. Such networks can be very useful in practice i.e. in the local detection of remote crucial events and the propagation of data reporting their realization to a control center. Typical sensor network scenarios involve scattering a large number of wireless nodes from an aircraft across an area of interest. The nodes then form a network through which collected data is routed to a base station. Adaptation is necessary to deal with the unpredictable network topologies that result from sensor node scatters and to manage resources (energy in particular) efficiently in response to changing conditions and requirements.

Wireless Sensor Network could contain hundreds of sensors that collect and some cases pre-process data before it is send to central node for final processing. In most cases sensors are deployed in remote location without capability to replace battery. This means that **one of the key elements for distributed sensors is long lifetime covering both reliability and energy efficiency because the battery limits lifetime of the sensors**. Energy efficiency should be taken account is all design phases starting from system design e.g. topology down to physical implementation constrains. In the literature has shown that energy efficiency can be improved in various areas when designing wireless sensor network e.g. VLSI design, protocols and network topology.

Many wireless sensor network applications require information about the geographic location of each sensor node. Besides the typical application of correlating sensor readings with physical locations, approximate geographical localization is also needed for many sensor network applications. Manual recording and entering the positions of each sensor node is impractical for very large sensor networks. To address the problem of assigning an approximate geographic coordinate to each sensor node, many automated localization algorithms have been developed. In this project we have proposed to use the GPS info to find the location of the nodes in the network.

For many applications, we envision large static ad hoc networks consisting of hundreds, thousands, or even millions of inexpensive wireless sensor nodes that may be placed either regularly or irregularly. Wireless sensor networks can be exposed to highly dynamic and hostile environments, and therefore. they must be tolerant to the failure of individual nodes. Algorithms for wireless sensor networks must be distributed to prevent single points of failure, and self-organizing for scalable deployment.

Wireless sensor networks are bandwidth and energy constrained. Self-organization algorithms that minimize the number of message transmissions (and receptions) can help conserve energy and bandwidth. Protocols with low message overhead are preferable in tactical wireless sensor networks, since for example, statistical methods of cryptanalysis require a large number of samples to be successful. Therefore, low message complexity is a highly desirable property of self organization algorithms for wireless sensor networks.

The autonomous set up of addressing and routing in a large network requires the decomposition of the network into connected clusters. Bounding the cluster size is a requirement imposed by routing protocol complexity. Furthermore, resource

constraints or specific network architectures can impose limits on the cluster size. Thus, it is desirable to develop distributed algorithms that can form bounded size clusters. Moreover, the cluster sizes produced should be as close as possible to the specified bound in order to limit the total number of clusters. This can help the construction of a flatter hierarchy for efficient routing.

In this project, k-means clustering algorithm is used to organize the nodes into clusters. The Location information of the sensor nodes is collected from source and the cluster heads were elected/selected based on the locations of the active sensor nodes in network. For selecting a cluster head, k-means algorithm is used to find the cluster center. The nearby sensor nodes of a cluster-head then forward their data to sink only via the cluster-head. For hop to hop packet forwarding, AODV protocol is used. That is to forward a packet from a sensor node to a cluster-head or cluster head to sink; normal routing protocols may be used.

The proposed clustering based mechanism reduces the number of management nodes in a large-scale sensor network. All the simulations of the proposed idea are simulated on ns2 network simulator and the performance of the proposed scheme is evaluated. Hence, clustering sensor nodes into small groups is an effective technique to achieve scalability, self-organization, power saving, channel access, routing, etc.

1.2 Design Challenges

In typical wireless sensor network applications, a large number of sensor and actuator devices (nodes) are scattered by an aircraft across a target area. A high power base station is then used by an operator to control the behavior of nodes and receive data from the nodes via a wireless network. Such applications pose a number of design challenges for sensor networks:

- Nodes are commonly deployed from an aircraft in a rapid, ad hoc manner, reventing designers from making assumptions about where particular nodes will be located, what the network topology will be, and the role of each node in the topology.
- Application requirements and environmental conditions often change during network operation.
- Nodes often fail during network operation, due to depleted energy, destruction, or movement out of transmission range.
- Given that the nodes are wireless and ideally quite small, they have very limited energy, most of which is consumed by data transmission.
- The large number of nodes in sensor networks, their limited lifetime, and their inherent disposability require a low per unit cost.

1.3 Routing in Sensor networks

There are a few inherent limitations of wireless or wired media such as low bandwidth, error prone transmissions, need for collision free channel access etc. These wireless nodes also have only a limited amount of energy available to them, since they derive energy from a personal battery and not from a constant power supply. Furthermore, since these sensor nodes are deployed in places where it is difficult to replace the nodes or their batteries, it is desirable to increase the longevity of the

network. Also preferably all the nodes should die together so that we can replace all the nodes simultaneously in the whole area.

Finding individual dead nodes and then replacing those nodes selectively would require pre-planned deployment and eliminate some advantages of these networks. Thus, the protocols designed for these networks must strategically distribute the dissipation of energy, which also increases the average life of the overall system.

In ad hoc networks all nodes are mobile and can be connected dynamically in an arbitrary manner. All nodes of these networks behave as routers and take part in discovery and maintenance of routes to other nodes in the network. Ad hoc networks are very useful in emergency search-and-rescue operations, meetings or conventions in which persons wish to quickly share information, and data acquisition operations in inhospitable terrain.These ad-hoc routing protocols can be divided into two categories:

1. Table-driven routing protocols.

In table driven routing protocols, consistent and up-to-date routing information to all nodes is maintained at each node.

2. On-Demand routing protocols.

In On-Demand routing protocols, the routes are created as and when required. When a source wants to send to a destination, it invokes the route discovery mechanisms to find the path to the destination.

In recent years, a variety of new routing protocols targeted specifically at this environment have been developed. There are four multi-hop wireless ad hoc network routing protocols that cover a range of design choices:

- Destination-Sequenced Distance-Vector (DSDV)

- Temporally Ordered Routing Algorithm (TORA)
- Dynamic Source Routing (DSR)
- Ad Hoc On-Demand Distance Vector Routing (AODV).

While DSDV is a table-driven routing protocol, TORA, DSR, AODV, fall under the On-demand routing protocols category.

1.3.1 Query Classification in Sensor Networks

Before discussing routing protocols for sensor networks, let us first categorize the different kinds of queries that can be posed to a sensor network. Based on the temporal property of data, i.e. whether user is interested in data collected in the past, the current snapshot view of the target regions or sensor-data to be generated in future for a given interval of time, queries can be classified as follows.

Historical queries: This type of query is mainly used for analysis of historical data stored at a remote base station or any designated node in the network in the absence of a base station. For example, "What was the temperature 2 hours back in the northwest quadrant?" The source nodes need not be queried to obtain historical data as it is usually stored outside the network or at a node equidistant from all anticipated sinks for that data.

- **One-time query:** One-time or snapshot queries provide the instantaneous view of the network. For example, "What is the temperature in the northwest quadrant now?" The query triggers a single query response; hence data traffic generated by one time queries is the least. These are usually time critical as user wants to be notified immediately about the current situation of the network. A

warning message that informs the user of some unusual activity in the network is an example of one-time query response that is time-critical.

- **Persistent Queries:** Persistent or long running queries, are mainly used to monitor a network over a time interval with respect to some parameters. For example, "the temperature in the northwest quadrant for the next 2 hours". A persistent query generates maximum query responses in the network depending on it's duration. The purpose of the persistent query is to perform periodic background monitoring. Energy efficiency is often traded with delay in response time of persistent queries to maximize utilization of network resources, as they are usually non critical.

1.3.2 Characteristics of Routing Protocols for Sensor Networks

For wired sensor networks, we can adopt the standards on existing routing protocols available for wired network. Traditional routing protocols defined for wireless ad-hoc networks are not well suited for wireless sensor networks due to the following reasons:

- Sensor networks are data centric. Traditional networks usually request data from a specific node, but sensor networks request data based on certain attributes such as "which area has temperature > 100°F?"

- In traditional wired and wireless networks, each node is given a unique id, which is used for routing. This cannot be effectively used in sensor networks because being data centric they do not require routing to and from specific nodes. Also, the large number of nodes in the network

implies large ids, which might be substantially larger than the actual data being transmitted.

- Adjacent nodes may have similar data. So instead of sending data separately from each node to the requesting node, it is desirable to aggregate similar data before sending it.

- The requirements of the network change with the application and hence, it is application-specific. For example, some applications need the sensor nodes to be fixed and not mobile while others may need data based only on one selected attribute (i.e., here the attribute is fixed).

Thus, sensor networks need protocols which are application specific, data-centric, capable of aggregating data and minimizing energy consumption.

1.4 Network Architecture

The two main architecture alternatives used for data communication in a sensor network are the hierarchical and the flat network architectures. The hierarchical network architecture is energy efficient for collecting and aggregating data within a large target region, where each node in the region is a source node. Hence hierarchical network protocols are used when data is to be collected from the entire sensor network. Flat network architecture is more suitable for transferring data between a source destination pair separated by a large number of hops.

1.4.1 Hierarchical Network Architecture

One way of minimizing the data transmissions over long distances is to cluster the network so that signaling/control overheads can be reduced, while critical functions such as media access, routing, and connection setup could be improved. While all nodes typically function as switches/routers, one node in each cluster is designated as the cluster head (CH), and traffic between nodes of different clusters must always be routed through their respective CHs or gateway nodes that are responsible for maintaining connectivity among neighboring CHs. The number of tiers within the network can vary according to the number of nodes, resulting in hierarchical network architecture as shown in Figure 3.

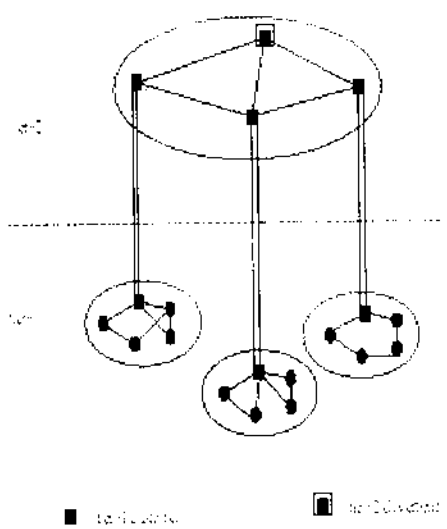


Figure 1.1: Two-tier sensor network architecture

Figure 3 shows two tiers of clusterhead where the double lines represent that CHs of tier-1 are cluster-members of the cluster at the next higher level that is tier-2. A proactive clustering algorithm for sensor networks, called LEACH is one of the initial data gathering protocols introduced by MIT's researchers Heinzelman et. al. . Each cluster has a CH that periodically collects data from its cluster members, aggregates it

and sends it to an upper level CH. Only the CH needs to perform additional data computations such as aggregation, etc. and the rest of the nodes sleep unless they have to communicate with the CH. In order to evenly distribute this energy consumption, all the nodes in a neighborhood take turns to become the CH for a time interval called the cluster period.

1.4.2 Flat Network Architecture

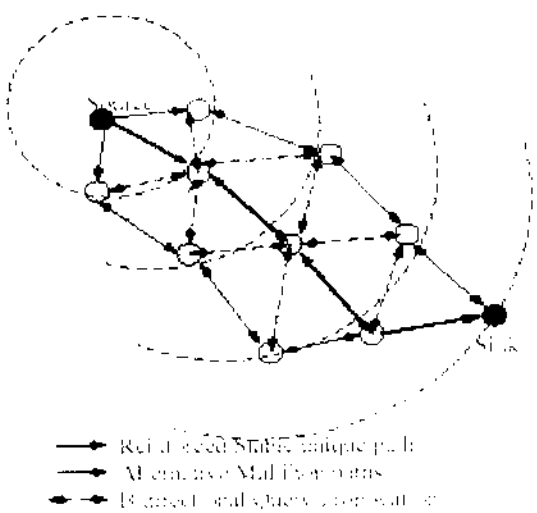


Figure 1.2: A flat sensor network

In flat network architecture as shown in Figure 4, all nodes are equal and connections are setup between nodes that are in close proximity to establish radio communications, constrained only by connectivity conditions and security limitations. Route discovery can be carried out in sensor networks using flooding that do not require topology maintenance as it is a reactive way of disseminating information. In flooding, each node receiving data packets broadcasts till all nodes or the node at which the packet was originated gets back the packet. But in sensor networks, flooding is minimized or avoided as nodes could receive multiple or duplicate copies of the same data packet due to nodes having common neighbors or sensing similar data. Intanagonwiwat et. al. have introduced a data dissemination paradigm called directed

diffusion for sensor networks, based on a flat topology. The query is disseminated (flooded) throughout the network with the querying node acting as a source and gradients are setup towards the requesting node to find the data satisfying the query. As we can observe from figure above, the query is propagated towards the requesting node along multiple paths shown by dashed lines. The arcs show how the query is directed towards the event of interest similar to a ripple effect. Events (data) start flowing towards the requesting node along multiple paths. To prevent further flooding, a small number of paths can be reinforced (shown by dark lines in the figure), among a large number of paths initially explored to form the multihop routing infrastructure so as to prevent further flooding. One advantage of the flat networks is the ease of creating multiple paths between communicating nodes, thereby alleviating congestion and providing robustness in the presence of failures.

1.5 Classification of Sensor Networks

Sensor networks can be classified into two types based on their mode of operation or functionality and the type of target applications:

Proactive Networks: In this scheme the nodes periodically switch on their sensors and transmitters, sense the environment and transmit the data of interest. Thus, they provide a snapshot of the relevant parameters at regular intervals and are well suited for applications that require periodic data monitoring.

Reactive Networks: In this scheme the nodes react immediately to sudden and drastic changes in the value of a sensed attribute and are well suited for time critical applications.

Once we have a network, we have to come up with protocols which efficiently route data from the nodes to the users, preferably using a suitable MAC (Medium Access Control) sub-layer protocol to avoid collisions.

2 DESIGN

2.1 Architecture

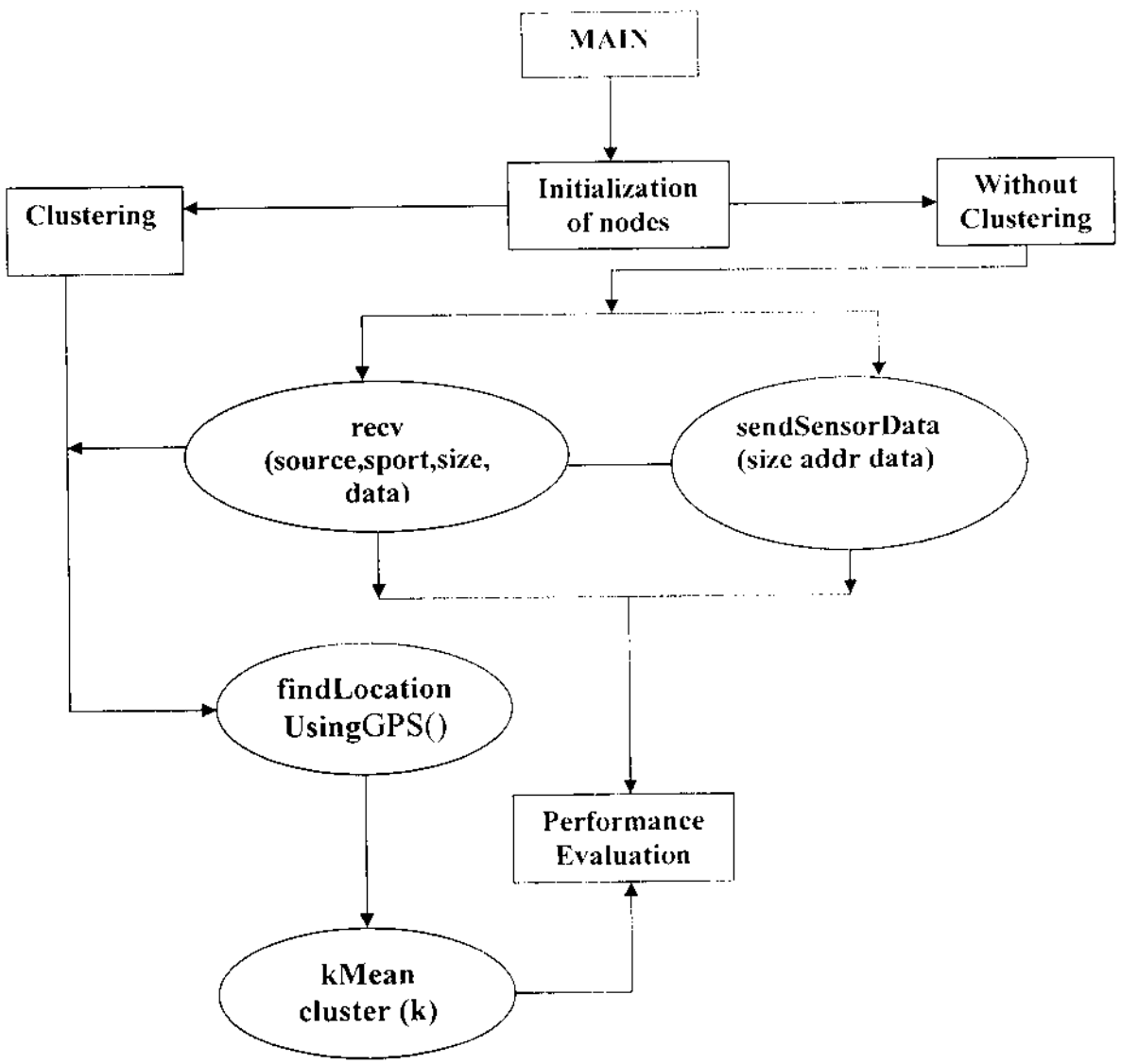


Figure 2.1: Architecture Block Diagram

2.2 Find Location Using GPS Info

Initially the position or location information of the nodes in the sensor network is unknown to the sink as well as the cluster heads. Hence after initialization of the nodes all the sensor nodes in the sensor network scenario broadcast their location information to every other node in the network. Hence after the creation of the cluster heads, the nodes in a particular cluster will forward data only to the cluster head and not to the sink. The cluster head will thus forward data to the sink.

2.3 Create Clusters and find Cluster Centers

This method is used to create k number of clusters from the given sensor network scenario. First Select k Center in the problem space or it can also be a random point. Partition the data into k clusters by grouping points that are closest to those k centers. Use the mean of these k clusters to find new centers. The process is repeated until the centers do not change. Then find the nearby central nodes from the calculated cluster centers.Make the central nodes as a cluster head. Now, the Cluster heads will announce its presence to all its neighbors. The neighboring nodes in turn will forward the data to sink via the cluster-head. The sink will periodically update the cluster-heads.

2.4 Receive Sensor Data

All the sensor nodes act as intermediate nodes in a multihop network .Hence they can either receive data or forward them to the other nodes in the broadcast range. Since we are going to the implement he concept of clustering, all nodes do not behave the same way. If the node is not a cluster head it will receive data and forward it to the neighboring nodes. If the node is a cluster head then the data is forwarded to the sink.

2.5 Send Sensor Data

The sensor nodes broadcast data periodically to the sink. In a scenario without clustering the intermediate nodes forward packets to their neighboring nodes and thus data is forwarded to the sink.In a clustering scenario,the cluster heads collect data from all the nodes in that particular cluster and data fusion takes place.The cluster head holds the data for a specified time thus improving network traffic.The time delay is random to avoid collisions.

3 CLUSTERING

3.1 Clustering Architecture

The proposed mechanism reduces the number of management nodes in a large-scale sensor network. One of the best methods is to develop hierarchical architecture and apply the 20/80 rule. Figure 1 illustrates the architecture of clustering networks

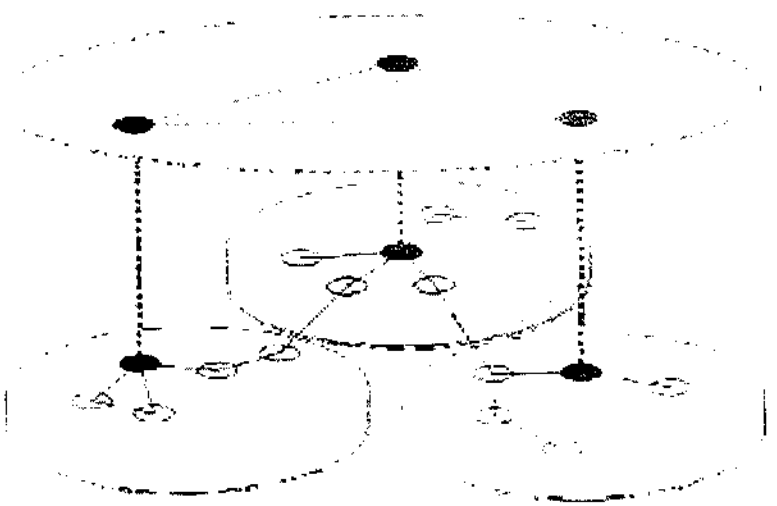


Figure 3.1: Architecture of clustering networks

3.2 The Clustering Problem

The following are the requirements that must be met for any clustering algorithm:

- Clustering is completely distributed. Each node independently makes its decisions based on local information.
- Clustering terminates within a fixed number of iterations(regardless of network diameter).
- At the end of each *TCP*, each node is either a cluster head, or a non-head node (which we refer to as regular node) that belongs to exactly one cluster.

- Clustering should be efficient in terms of processing complexity and message exchange.
- Cluster heads are well-distributed over the sensor field

3.3 Stages in a Clustering Task

A pattern (or feature vector) is a single data item that is used by clustering algorithm. It typically consists of a vector of d measurements (where d the dimensionality of the data):

$$\mathbf{x} = (x_1, \ldots, x_d)$$

The individual scalar components x_i *of* a patter X is are called features (or attributes). Pattern representation refers to the number of classes, the number of available patterns, the number, type and scale of the features available to clustering algorithms.

Feature selection is the process of identifying the most effective subset of original features to use in clustering. Feature extraction is the use of one or more transformations of the input features to produce new salient features. Either or both of these techniques can be used to obtain what is called a *feature set* (or feature vector).

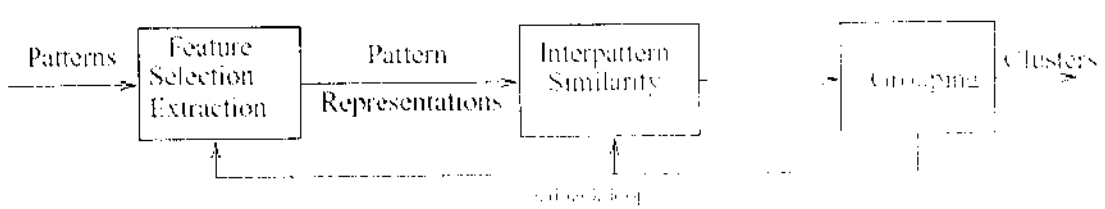


Figure 3.2: Stages in Clustering

Pattern proximity is usually measured by a distance function defined on pairs of patterns. A variety of distance functions are in use in various communities. A simple distance measure can often be used to reflect dissimilarity between two patterns, where

other similarity measures can be used to characterize the conceptual similarity between two patterns. The Euclidian distance metric can be defined as follows:

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2 \right)^{1/2}$$

$$= \|\mathbf{x}_i - \mathbf{x}_j\|_2,$$

where X_1 and X_2 are two patterns. Euclidian distance metric works well when the data set has"compact" or "isolated" clusters.

Another class of metrics characterizes conceptual similarity between two patterns. For example in Conceptual cluster (which we don't discuss in this paper), the similarity between X_1, X_2 is defined as

$$s(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i, \mathbf{x}_j, \ell, \mathcal{C}),$$

where $\mathcal{C}$ is a set of pre-defined concepts.

The Grouping step represents the organization of patterns into clusters based on pattern similarity. There are many clustering methods available, and each of them may give a different grouping of a dataset. The choice of a particular method will depend on the type of output desired, the known performance of method with particular types of data, the hardware and software facilities available and the size of the dataset. The taxonomy of clustering algorithms can be seen in the figure3.3.

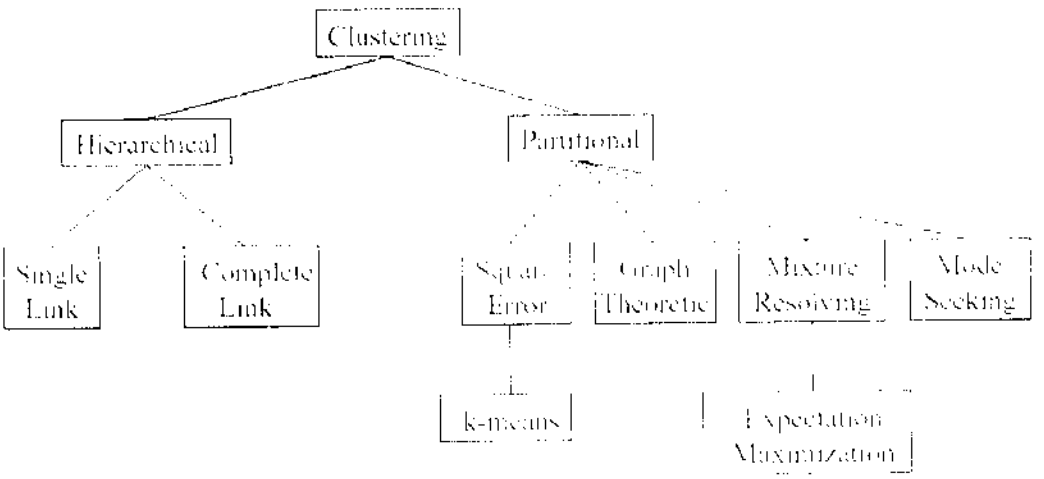


Figure 3.3: Taxonomy of clustering approaches

3.4 Naive k-means algorithm

One of the most popular heuristics for solving the k-means problem is based on a simple iterative scheme for finding a locally optimal solution. This algorithm is often called the k-means algorithm. There are a number of variants to this algorithm, so to clarify which version we are using, we will refer to it as the naive k-means algorithm as it is much simpler compared to the other algorithms described here. This algorithm is also referred to as the Lloyd's algorithm in [4].

The naive k-means algorithm partitions the dataset into 'k' subsets such that all records, from now on referred to as points, in a given subset "belong" to the same center. Also the points in a given subset are closer to that center than to any other center. The partitioning of the space can be compared to that of Voronoi partitioning except that in Voronoi partitioning one partitions the space based on distance and here we partition the points based on distance.

The algorithm keeps track of the centroids of the subsets, and proceeds in simple iterations. The initial partitioning is randomly generated, that is, we randomly initialize

the centroids to some points in the region of the space. In each iteration step, a new set of centroids is generated using the existing set of centroids following two very simple steps. Let us denote the set of centroids after the i^{th} iteration by $C^{(i)}$. The following operations are performed in the steps:

(i) Partition the points based on the centroids C(i), that is, find the centroids to which each of the points in the dataset belongs. The points are partitioned based on the Euclidean distance from the centroids.

(ii) Set a new centroid $c(i+1) \in C(i+1)$ to be the mean of all the points that are closest to $c(i) \in C(i)$ The new location of the centroid in a particular partition is referred to as the new location of the old centroid.

The algorithm is said to have converged when recomputing the partitions does not result in a change in the partitioning. In the terminology that we are using, the algorithm has converged completely when $C^{(i)}$ and $C^{(i+1)}$ are identical. For configurations where no point is equidistant to more than one center, the above convergence condition can always be reached. This convergence property along with its simplicity adds to the attractiveness of the k-means algorithm.

The naive k-means should perform a large number of "nearest-neighbor" queries for the points in the dataset. If the data set is 'd' dimensional and there are 'N' points in the dataset, the cost of a single iteration is O(kdN). As one would have to run several iterations, it is generally not feasible to run the naive k-means algorithm for large number of points.

Sometimes the convergence of the centroids (i.e. $C^{(i)}$ and $C^{(i+1)}$ being identical) takes several iterations. Also in the last several iterations, the centroids move very little. As running the expensive iterations so many more times might not be efficient, a

measure of convergence of the centroids is needed so that the iterations are stopped once the convergence criterion is met. Distortion is the most widely accepted measure.

3.5 Clustering Error

Clustering error measures the same criterion and is sometimes used instead of distortion. In fact k-means algorithm is designed to optimize distortion. Placing the cluster center at the mean of all the points minimizes the distortion for the points in the cluster. Also when another cluster center is closer to a point than its current cluster center, moving the cluster from its current cluster to the other can reduce the distortion further. The above two steps are precisely the steps done by the k-means cluster. Thus k-means reduces distortion in every step locally. The k-Means algorithm terminates at a solution that is locally optimal for the distortion function. Hence, a natural choice as a convergence criterion is distortion. Among other measures of convergence used by other researchers, we can measure the sum of Euclidean distance of the new centroids from the old centroids as in [9]. In this thesis we always use clustering error/distortion as the convergence criterion for all variants of k-means algorithm.

Definition 1: Clustering error is the sum of the squared Euclidean distances from points to the centers of the partitions to which they belong.

Mathematically, given a clustering ϕ, we denote by $\phi(x)$ the centroid this clustering associates with an arbitrary point x (so for k-means, $\phi(x)$ is simply the center closest to x). We then define a measure of quality for ϕ:

$$distortion_{\phi} = \frac{1}{N}\sum_{x} |x - \phi(x)|^2$$

Where |a| is used to denote the norm of a vector 'a'. The lesser the difference in distortion over successive iterations, the more the centroids have converged. Distortion is therefore used as a measure of goodness of the partitioning.

3.6 Problems with K-Means

The algorithm is simple and has nice convergence but there are a number of problems with this

1. Selection of value of K is itself an issue and sometimes its hard to predict before hand the number of clusters that would be there in data.
2. Experiments have shown that outliers can be a problem and can force algorithm to identify false clusters.

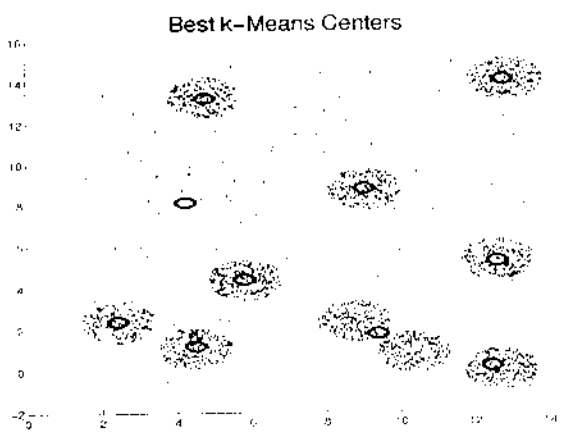


Figure 3.4: Identification of false cluster due to outliers

4 NETWORK SIMULATOR

Various simulators commonly available are:

- OPNET
- NS –2
- GloMoSim/Parsec

In our project we have used NS-2 as the simulator and used TCL/Shell Scripting as the programming languages.

4.1 NS –2 Introduction

NS-2 is a discrete event network simulator that has begun in 1989 as a variant of the REAL network simulator. Initially intended for wired networks, the Monarch Group at CMU have extended NS-2 to support wireless networking such as MANET and wireless LANs as well. Most MANET routing protocols are available for NS-2, as well as an 802.11 MAC layer implementation.

NS-2's code source is split between C++ for its core engine and OTcl, an object oriented version of TCL for configuration and simulation scripts. The combination of the two languages offers an interesting compromise between performance and ease of use.

Implementation and simulation under NS-2 consists of 4 steps:

- Implementing the protocol by adding a combination of C++ and OTcl code to NS-2's source base;
- Describing the simulation in an OTcl script;
- Running the simulation and
- Analyzing the generated trace files.

Implementing a new protocol in NS-2 typically requires adding C++ code for the protocol's functionality, as well as updating key NS-2 OTcl configuration files in order for NS -2 to recognize the new protocol and its default parameters. The C++ code also describes which parameters and methods are to be made available for OTcl scripting. The NS-2 architecture follows closely the OSI model. We have adapted the implementation of flooding provided in NS-2 in the context of diffusion in sensor networks. An agent in NS-2 terminology represents an endpoint where network packets are constructed, processed or consumed. Such an Agent was implemented at the Application layer for the broadcast source, and the simulation trace was collected at the MAC layer.

4.2 Components of ns

1. Nam (Network Animator)

- Visualize ns (or other) output
- GUI input simple ns scenarios

2. Pre-processing

- Traffic and topology generators

3. Post-processing

- simple trace analysis, often in Awk, Perl, or Tcl

4.3 Goals of Ns

- Support networking research and education
  - Protocol design, traffic studies, etc.
  - Protocol comparison
- provide a collaborative environment
  - Freely distributed, open source
  - Share code, protocols, models, etc.
  - Allow easy comparison of similar protocols

- Increase confidence in results
- Models provide useful results in several Situations

- It covers multiple layers

  Application layer, transport layer, network layer and link layer.

- Supports the simulation of intserv/diffserv, Multicast, Transport, Applications, Wireless (fixed, mobile, satellite)

4.4 Important NS-2 Programming Languages

- **NS-2**

  It is an object oriented simulator, written in C++, with an *OTcl* (Object Tool Command Language) interpreter as a front-end.

- **Back-end C++**
  - Defining new agents, protocols and framework.
  - Manipulations at the byte/bit levels.
  - if you have to change the behaviour of an existing C++ class in ways that weren't anticipated

- **Front-end Otcl**
  - Topologies, scenarios, simulations, ...
  - Script language (easy topology modifications)
  - if you can do what you want by manipulating existing C++ objects

Why Two Languages

- Simulator had two distinct requirements
  - Detailed simulation of Protocol(Run-time speed)
  - Varying parameters or configuration(Change model & rerun)
- C++ is fast to run but slower to change
- Otcl runs much slower but can be changed quickly

4.5 ns Directory Structure

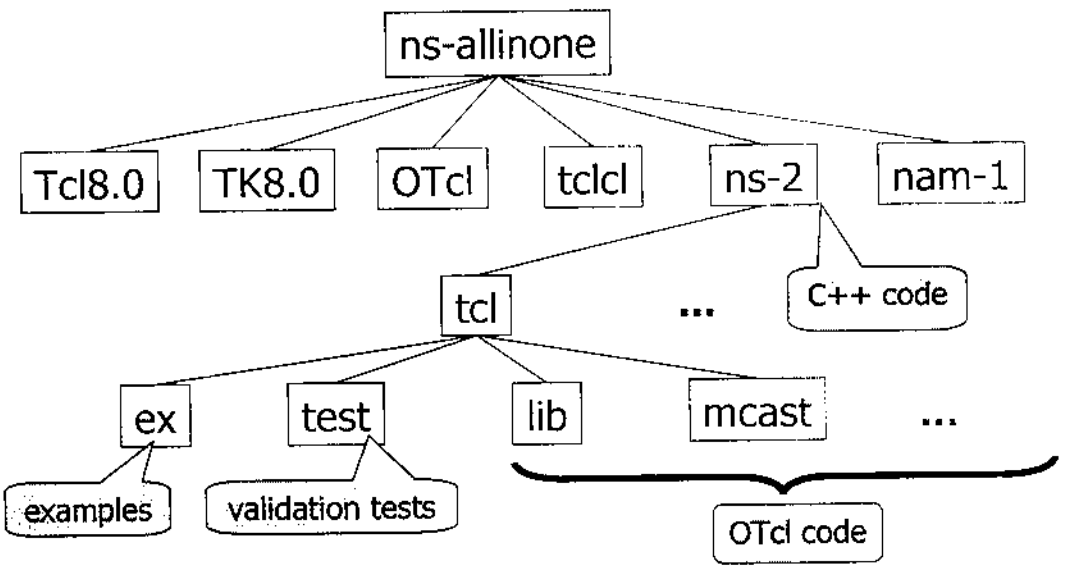


Figure 4.1: ns Directory Structure

4.6 Wireless Simulation

Example wireless program

Configuring 6 nodes and stabling tcp ,udp traffic between also checks movement of nod

Defining wireless option

```
set val(chan)    Channel/WirelessChannel  ;     # channel type
set val(prop)    Propagation/TwoRayGround ;     # radio-Propagation model
set val(ant)      Antenna/OmniAntenna  ;        # Antenna type
set val(ll)       LL  ;                         # Link layer type
set val(ifq)      Queue/DropTail/PriQueue  ;    # Interface queue type
set val(ifqlen)   50  ;                         # max packet in ifq
set val(netif)   Phy/WirelessPhy ;              # network interface type
set val(mac)      Mac/802_11 ;                  # MAC type
```

```
set val(rp)        DSDV ;                    # ad-hoc routing protocol
set val(nn)        6  ;                      # number of mobilenodes
set val(x)         500
set val(y)         500

set ns_    [new Simulator]
set tracefd     [open first.tr w]
$ns_ trace-all $tracefd
set namtrace [open first.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

Create & define topology
set topo     [new Topography]
$topo load_flatgrid $val(x) $val(y)


Create God – General Operations Descriptor
create-god $val(nn)


Configure node
      $ns_ node-config -adhocRouting $val(rp) \
                  -llType $val(ll) \
                  -macType $val(mac) \
                  -ifqType $val(ifq) \
                  -ifqLen $val(ifqlen) \
                  -antType $val(ant) \
                  -propType $val(prop) \
                  -phyType $val(netif) \
                  -topoInstance $topo \
                  -channelType $val(chan) \
                  -agentTrace ON \
                  -routerTrace ON \
                  -macTrace OFF \
                  -movementTrace OFF
```

```
for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node ]
        $node_($i) random-motion 0
    }

Adding nodes and setting positions
$node_(0) set X_ 20.0
$node_(0) set Y_ 105.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 60.0
$node_(1) set Y_ 205.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 100.0
$node_(2) set Y_ 305.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 140.0
$node_(3) set Y_ 105.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 180.0
$node_(4) set Y_ 205.0
$node_(4) set Z_ 0.0
$node_(5) set X_ 220.0
$node_(5) set Y_ 305.0
$node_(5) set Z_ 0.0

for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 50}
Adding agents and traffic source

set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
```

```
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 0.1 "$ftp start"
$ns_ at 0.4 "$ftp stop"
$ns_ at 0.2 "$node_(2) setdest 140.0 120.0 70.0"


set tcp1 [new Agent/TCP]

set sink1 [new Agent/TCPSink]

$ns_ attach-agent $node_(2) $tcp1
$ns_ attach-agent $node_(3) $sink1
$ns_ connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns_ at 0.2 "$ftp1 start"
$ns_ at 1.6 "$ftp1 stop"

#udp traffic between nodes

set udp [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp

set null [new Agent/Null]
$ns_ attach-agent $node_(5) $null
$ns_ connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

```
$cbr set packetSize_ 1000
$cbr set rate_ 0.01Mb
$cbr set random_ false

$ns_ at 0.3 "$cbr start"
$ns_ at 1.8 "$cbr stop"

Tell nodes when the simulation ends

$ns_ at 3.0001 "stop"
$ns_ at 3.0002 "puts \"NS EXITING...\" ; $ns   halt"

proc stop {} {
   global ns_ tracefd
   close $tracefd
     exec nam first.nam &
  exit 0
}
```

5 THE SIMULATION ENVIRONMENT

5.1 Hardware specifications.

The following hardware were used to simulate this algorithms

System Processor	:	Pentium IV @ 1.7 GHz
Main Memory	:	128 Mb
Video card	:	AGP with 32 Mb RAM
Hard Disk drive	:	10 GB Free Space

5.2 Software specifications.

The following software were used during developing this system:

Operating System	:	Red Hat Linux 9
Simulator	:	Ns2 version 2.28
Graph	:	X-Graph/gnuplot
Visualization	:	NAM (Network Animator)
Programming Language	:	Tcl, Shell Scripting

6 ALGORITHM ANALYSIS

6.1 K-means Algorithm

k-Means algorithm is very popular for data clustering. In this proposed algorithm, k-Means algorithm is used to cluster the nodes with respect to their locations in the MANET and select a central node in each cluster to make it as a cluster head.

1. From sink, resolve the locations of all the nodes in the network.
2. For this we can use a service like GPS or we can resolve them by using the agents in all the nodes which will provide the information on request. For the second approach simple flooding can be used for initial location discovery. (in this simulation, GPS was assumed)
3. Then Select k Center in the problem space (it can be random).
4. Partition the data into k clusters by grouping points that are closest to those k centers.
5. Use the mean of these k clusters to find new centers.
6. Repeat steps 3 and 5 until centers do not change.
7. Find the nearby central nodes from the calculated cluster centers.
8. Make the central nodes as a cluster head.
9. Now, the Cluster heads will announce its presence to all its neighbors.
10. The neighboring nodes in turn will forward the data to sink via the cluster-head.
11. The sink will periodically update the cluster-heads.

The k-means algorithm partitions the dataset into 'k' subsets such that all records, from now on referred to as points, in a given subset "belong" to the same center. Also the points in a given subset are closer to that center than to any other center. The algorithm keeps track of the centroids of the subsets, and proceeds in simple iterations. The initial partitioning is randomly generated, that is, we randomly

initialize the centroids to some points in the region of the space. In each iteration step, a new set of centroids is generated using the existing set of centroids. Let us denote the set of centroids after the i^{th} iteration by $C^{(i)}$. The following operations are performed in the steps:

(iii) Partition the points based on the centroids C(i), that is, find the centroids to which each of the points in the dataset belongs. The points are partitioned based on the Euclidean distance from the centroids.

(iv) Set a new centroid $c(i+1) \in C(i+1)$ to be the mean of all the points that are closest to $c(i) \in C(i)$ The new location of the centroid in a particular partition is referred to as the new location of the old centroid.

The algorithm is said to have converged when recomputing the partitions does not result in a change in the partitioning. In the terminology that we are using, the algorithm has converged completely when $C^{(i)}$ and $C^{(i+1)}$ are identical. For configurations where no point is equidistant to more than one center, the above convergence condition can always be reached. This convergence property along with its simplicity adds to the attractiveness of the k-means algorithm.

The k-means algorithm needs to perform a large number of "nearest-neighbor" queries for the points in the dataset. If the data is 'd' dimensional and there are 'N' points in the dataset, the cost of a single iteration is O(kdN). Sometimes the convergence of the centroids (i.e. $C^{(i)}$ and $C^{(i+1)}$ being identical) takes several iterations. Also in the last several iterations, the centroids move very little. As running the expensive iterations so many more times might not be efficient, we need a measure of convergence of the centroids so that we stop the iterations when the convergence criterion is met. Distortion is the most widely accepted measure

7 IMPLEMENTATION

The implementation is realized using the Network Simulator. The simulator has all the protocols implemented. The AODV protocol is chosen for using in the Ad-Hoc network we discuss. The protocol is very dynamic changes in the systems. The protocol uses 5 packet formats for doing its dynamic routing needs.

7.1 Handling Functions

The following are the methods that implement the wireless sensor network scenario with clustering

1. findLocationUsingGPSInfo ()
2. kMeanCluster (k)
3. recv (source,sport,size,data)
4. sendSensorData (size addr data)

7.2 Generating traffic and mobility models

Traffic models

Random traffic connections of TCP and CBR can be setup between mobile nodes using a traffic-scenario generator script. This traffic generator script is available under ~ns/indep-utils/cmu-scen-gen and is called cbrgen.tcl. It can be used to create CBR and TCP traffics connections between wireless mobilenodes. So the command line looks like the following:

ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections][-rate rate]

Mobility models

The node-movement generator is available under ~ns/indep-utils/cmu-scen-gen/setdest directory and consists of setdest{.cc,.h} and Makefile. The command would look like

./setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime] \ [-x maxx] [-y maxy] > [outdir/movement-file]

Simulations could be carried out with more complex scenarios using higher traffic rates. However, the simulations take much longer to complete and the trace file generated by each run ends up to few Mb of user space.

7.3 Parsing the Simulation trace files

After each simulation, trace files recording the traffic and node movements are generated. These files need to be parsed in order to extract the information needed to measure the performance metrics. The new trace format was used for parsing.

The new trace format looks like:

:67662078 -Hs 0 -Hd -1 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 –Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 –Ii 20 -Is 0.255 -Id -1.255 – It

Here, we see that a packet was sent (s) at time (t) 0.267662078 sec. from source node (Hs) 0 to destination node (Hd) 1. The source node id (Ni) is 0, it's x-co-ordinate (Nx) is 5.00, it's y-co-ordinate (Ny) is 2.00, it's z-co-ordinate (Nz) is 0.00, it's energy level (Ne) is 1.000000, the trace level (Nl) is RTR and the node event (Nw) is blank. The

MAC level information is given by duration (Ma) 0, destination Ethernet address (Md) 0,the source Ethernet address (Ms) is 0 and Ethernet type (Mt) is 0. The IP packet level information like packet id (Ii), source address.source port number is given by (Is) while the destination address.destination port number is (Id).

7.4 Performance Parameters

Calculating Packet delivery fraction (pdf):

It is the ratio between the number of data packets delivered to the member to that of the number of packets supposed to be received by it. This metric gives a measure about the packet loss. A high packet delivery ratio implies an efficient protocol.

$$\text{Packet delivery ratio (in \%)} = \frac{\text{Actual packets received}}{\text{Packets supposed to be received}} \times 100$$

Calculate the number of "sent packets" that have the trace form:

```
- Nl AGT.*-Is (\d{1,3})\.\d{1,3} -Id (\d{1,3})\.\d{1,3}.*-It
cbr.*-Ii (\d{1,6})/
```

AGT => Agent Level Trace

Calculate the number of "received packets" of the trace form:

```
-t (\d{1,3}\.\d{9}).*-Nl AGT.*-Is (\d{1,3})\.\d{1,3} -Id
(\d{1,3})\.\d{1,3}.*-It cbr.*-Ii (\d{1,6})/
```

Calculating Normalized routing load:

Normalized routing load = (routing packets sent) / receives.

Calculate the routing packet sent from the trace file:

-Nl RTR.-It (?:AODV|DSR|message) -Il (\d{1,4})/

f=> forward

RTR=> Routing Trace Level

7.5 Source Code

 # Algorithm I(Scenario without Clustering)

```
#####################################################################
#########  Sensor Network Simulation Without Clustering  ########
#####################################################################

Mac/Simple set bandwidth_ 1Mb

set SENSOR_AGENT_PORT 42

set PercentageOfActiveDataSenders    75        ; # 1 to 100


set NamAnimationSpeed        250u   ;#in Micro Seconds
set SensorDataSize           64     ;# in bytes Max 1500
set SensorDataInterval       1      ;#in Seconds
set ChannelErrorRate         0.15
set num_nodes                50
set TotalSimulationTime      10
# variables which control the number of nodes and how they're grouped
# (see topology creation code below)

set val(chan)        Channel/WirelessChannel    ;#Channel Type
set val(prop)        Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)       Phy/WirelessPhy            ;# network interface type
set val(mac)         Mac/802_11                 ;# MAC type
set val(ifq)         Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)          LL                         ;# link layer type
set val(ant)         Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)      50                         ;# max packet in ifq
set val(rp)          AODV

# size of the topography
set val(x)           700
set val(y)           500

set ns [new Simulator]
```

```
set f [open Algorithm1.tr w]
$ns trace-all $f
set nf [open Algorithm1.nam w]

$ns namtrace-all-wireless $nf $val(x) $val(y)

$ns use-newtrace

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

#
# Create God
#
create-god $num_nodes


$ns color 0 blue
$ns color 1 red
$ns color 2 chocolate
$ns color 3 red
$ns color 4 brown
$ns color 5 tan
$ns color 6 gold
$ns color 7 black

set chan_1_ [new $val(chan)]

proc ChannelError {} {
global ChannelErrorRate
   set err [new ErrorModel]
   $err unit packet
   $err set rate_ $ChannelErrorRate
   #$err ranvar [new RandomVariable/Uniform]
   $err drop-target [new Agent/Null]
   return $err
}


$ns node-config -adhocRouting $val(rp) \
```

```
        -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace ON \
        -movementTrace OFF \
        -IncomingErrProc ChannelError\
        -channel $chan_1_

# subclass Agent/MessagePassing to make it do flooding


Class Agent/SensorAgent -superclass Agent/MessagePassing

set rng [new RNG]
$rng seed 0


Agent/SensorAgent instproc recv {source sport size data} {
  $self instvar node_ SentBytes RecievedBytes IsClusterHead FusedDataSize
  global ns PercentageOfActiveDataSenders SensorDataSize
  set rng [new RNG]
  $rng seed 0

    if {[$node_ node-addr]==0 } {

     set RecievedBytes [expr $RecievedBytes + $SensorDataSize]
  }


}
```

```
Agent/SensorAgent instproc SendSensorData {size addr data} {
   $self instvar node_
   global ns SENSOR_AGENT_PORT

   $self sendto $size $data $addr $SENSOR_AGENT_PORT

}

#the following proceedure is to send Sensor Data Periodically
Agent/SensorAgent instproc SheduleSensorDataTransmission {size} {
$self instvar node_ ClusterHead SentBytes
global ns SensorDataInterval

   set now [$ns now ]
   #$ns trace-annotate "$now : [$node_ node-addr] is sending Data"
   #puts "$now : [$node_ node-addr] is sending Data"

   set Data "DummySensorData"

   $ns at $now "$self SendSensorData $size $ClusterHead $Data"
   set SentBytes [expr $SentBytes + $size]
set rng [new RNG]
$rng seed 0

   #Re-Schedule The Procedure
   set rr [$rng integer 10]
   set BroadcastJitter [expr 0.99/ ($rr.0+1) ]
   #puts $BroadcastJitter
   $ns at [expr $now + $SensorDataInterval + $BroadcastJitter ] "$self
SheduleSensorDataTransmission $size"
}

#Create Sensor Nodes
for {set i 0} {$i < $num_nodes } {incr i} {
     set x [expr [ $rng integer ($val(x)-150)] +125 ]
     set y [expr [ $rng integer ($val(x)-50)] +25 ]
     set SensorNode($i) [$ns node]
     #[lindex [Node($ni) array get netif_] 1] set Pt_ 0.1 ;# to limite the Transmission
Range around/below 100m

     $SensorNode($i) set X_ $x
     $SensorNode($i) set Y_ $y
```

```
    $SensorNode($i) set Z_ 0.0
    $SensorNode($i) color black

    #$ns at 0.0 "$Node($i) setdest $x.0 $y.0 3000.0"

    set SensorAgent($i) [new Agent/SensorAgent]
    $SensorNode($i) attach $SensorAgent($i) $SENSOR_AGENT_PORT
    $SensorAgent($i) set IsClusterHead "false"
    $SensorAgent($i) set ClusterHead 0
    $SensorAgent($i) set SentBytes 0
    $SensorAgent($i) set RecievedBytes 0
    $SensorAgent($i) set FusedDataSize 0

    if { [$rng integer 100]<$PercentageOfActiveDataSenders} {
      $ns at 0.1 "$SensorAgent($i) SheduleSensorDataTransmission
$SensorDataSize "
    }
}

$SensorNode(0) color red
$SensorNode(0) set X_ 20
$SensorNode(0) set Y_ [expr $val(y) / 2 +100 ]

for {set i 0} {$i < $num_nodes } {incr i} {
    $ns initial_node_pos $SensorNode($i) 20
}

#$ns at 0.00001 "$SensorNode(0) setdest 20.0 20.0 30000.0"
$ns at 0.0 "$SensorNode(0) color red"
$ns at 0.0 "$SensorNode(0) label BS/Sink"



$ns at $TotalSimulationTime.1 "finish"

proc finish {} {
    global ns f nf val NamAnimationSpeed num nodes SensorAgent
TotalSimulationTime
    $ns flush-trace
    close $f
    close $nf
    exec nam -r $NamAnimationSpeed Algorithm1.nam &
```

```
    exec ns graph.tcl &
    set TotalSentBytes 0.0
    set TotalRecievedBytes 0.0
    for {set i 0} {$i < $num_nodes } {incr i} {
      set TotalSentBytes  [expr $TotalSentBytes + | $SensorAgent($i) set
SentBytes| ]
    }
    set TotalRecievedBytes  [$SensorAgent(0) set RecievedBytes ]



    puts "Total Sent Bytes : $TotalSentBytes "
    puts "Total Recieved Bytes : $TotalRecievedBytes "
    puts "The Delivery Ratio : [expr $TotalRecievedBytes /$TotalSentBytes *100.0]"

    exec ns graph.tcl &

    #set Throughput  [expr
$TotalRecievedBytes.0/$TotalSimulationTime.0*8.0/1000000.0]
    #puts "The Average Throughput : $Throughput"


    exit 0
}
$ns run
```

 # Algorithm II(Scenario with Clustering)

```
#################################################################
#########     Clustered Sensor Network Simulation     ########
#################################################################

Mac/Simple set bandwidth_ 1Mb

set SENSOR_AGENT_PORT 42

set PercentageOfActiveDataSenders    75        ; # 1 to 100


set NamAnimationSpeed        250u   ;#in Micro Seconds
set SensorDataSize           64     ;# in bytes Max 1500
```

```
set MaxFusedDataSize          512    ;# The Maximum size of the Data which a cluster
head can collect and send
set SensorDataInterval        1      ;#in Seconds
set ChannelErrorRate          0.15
set num_nodes                 50
set TotalSimulationTime       10

# variables which control the number of nodes and how they're grouped
# (see topology creation code below)



set val(chan)         Channel/WirelessChannel    ;#Channel Type
set val(prop)         Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)        Phy/WirelessPhy            ;# network interface type
set val(mac)          Mac/802_11                 ;# MAC type
set val(ifq)          Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)           LL                         ;# link layer type
set val(ant)          Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)       50                         ;# max packet in ifq
set val(rp)           AODV

# size of the topography
set val(x)            700
set val(y)            500


set ns [new Simulator]

set f [open Algorithm2.tr w]
$ns trace-all $f

set nf [open Algorithm2.nam w]

$ns namtrace-all-wireless $nf $val(x) $val(y)

$ns use-newtrace

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)
```

```
#
# Create God
#
create-god $num_nodes


$ns color 0 blue
$ns color 1 red
$ns color 2 chocolate
$ns color 3 red
$ns color 4 brown
$ns color 5 tan
$ns color 6 gold
$ns color 7 black

set chan_1_ [new $val(chan)]

proc ChannelError {} {
global ChannelErrorRate
   set err [new ErrorModel]
   $err unit packet
   $err set rate_ $ChannelErrorRate
   #$err ranvar [new RandomVariable/Uniform]
   $err drop-target [new Agent/Null]
   return $err
}


$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace OFF \
```

```
        -IncomingErrProc ChannelError\
        -channel $chan_1_

# subclass Agent/MessagePassing to make it do flooding


Class Agent/SensorAgent -superclass Agent/MessagePassing

set rng [new RNG]
$rng seed 0


Agent/SensorAgent instproc FindLocationUsingGPSInfo {} {

global ns SensorNode Location num_nodes

for {set i 0} {$i < $num_nodes} {incr i} {
   set Location($i,0) [$SensorNode($i) set X_ ]
   set Location($i,1) [$SensorNode($i) set Y_ ]
   #puts "$Location($i,0) "
}
}


Agent/SensorAgent instproc kMeanCluster {k} {
$self instvar node_
global ns SensorNode Location num_nodes rng twidth SensorAgent


   #assign initial same cluster labels for all nodes
   for {set i 0} {$i < $num_nodes} {incr i} {
      set cluster($i) 0
   }

   #randomly select initial cenres
   for {set i 0} {$i < $k} {incr i} {
      for {set j 0} {$j < 2} {incr j} {
      set mean($i,$j) [expr [$rng integer 750 ]]
      #puts $mean($i,$j)
```

```
    }
  }
  set flips 1

  while { $flips > 0 } {
    set flips 0

    for {set j 0} {$j < $k} {incr j} {
      set count($j) 0
      for {set i 0} {$i < 2} {incr i} {
        set sum($j,$i) 0.0;
      }
    }


    for {set i 0} {$i < $num_nodes} {incr i} {
      set dmin 99999999999.9
      set ccolor $cluster($i)

      for {set j 0} {$j < $k} {incr j} {
        set dx 0.0
        for {set c 0} {$c < 2} {incr c} {
            #puts "$dx $Location($i,$c) $mean($j,$c)"
            set dx [expr $dx +( ($Location($i,$c) - $mean($j,$c)) * (
$Location($i,$c) - $mean($j,$c) ))]
          }

        if { $dx < $dmin } {
          set ccolor $j
          set dmin $dx
        }
      }

      if { $cluster($i) != $ccolor } {
        incr flips
        set cluster($i) $ccolor
      }


      set count($cluster($i)) [expr $count($cluster($i)) + 1]

      for {set j 0} {$j < 2} {incr j} {
```

```
            set sum($cluster($i),$j) [ expr $sum($cluster($i),$j) + SLocation($i,$j)]
        }
    }

    for {set i 0} {$i < $k} {incr i} {
        for {set j 0} {$j < 2} {incr j} {
        if { $count($i) >0 } {
        set mean($i,$j) [expr $sum($i,$j) / $count($i)]
        }
        }
    }

}


for {set i 0} {$i < $num_nodes} {incr i} {
#puts $cluster($i)

$SensorAgent($i) set ClusterID $cluster($i)

switch $cluster($i) {
  0 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color blue"
  }
  1 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color cyan"
  }
  2 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color chocolate"
  }
  3 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color brown"
  }
  4 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color tan"
  }
  5 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color gold"
  }
  6 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color green"
```

```
  }
  7 {
  $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color yellow"
  }
  default {
    $ns at [expr [$ns now]+ 0.011] " $SensorNode($i) color black"
  }

} ;# end of switch


\$SensorAgent($i) set IsClusterHead "false"
} ;# end of for

$ns at [expr [$ns now]+ 0.011] " $SensorNode(0) color red"

puts "Marking Clusters and Cluster Centres .."
puts "The Calculated Cluster Centres"

for {set i 0} {$i < $k} {incr i} {
   if {$count($i) > 0 } {
   set x  [expr round($mean($i,0))]
   set y  [expr round($mean($i,1))]
    puts "$x $y"
   }
   set centre($i,0) 0
   set centre($i,1) 0
}


puts "The node Locations nearby the centres"

for {set j 0} {$j < $k} {incr j} {

   if {$count($j) > 0 } {
      set dmin 99999999999.9
      for {set i 0} {$i < $num_nodes} {incr i} {
         if {$cluster($i) == $j } {

            set d1 [expr (($mean($j,0) -  $Location($i,0)) * ($mean($j,0) -
$Location($i,0)))]
```

```
            set d2 [expr (($mean($j,1) - $Location($i,1)) * ($mean($j,1) -
$Location($i,1)))]
            set d [expr sqrt($d1 + $d2)]
            if {$d < $dmin} {
               set centre($j,0) $Location($i,0)
               set centre($j,1) $Location($i,1)
               set tn $i
               set dmin $d
            }
         }
      }
      for {set i 0} {$i < $num_nodes} {incr i} {
         if {$cluster($i) == $j } {
         $SensorAgent($i) set ClusterHead $tn
         }
      }
      puts "Node $tn : ( $centre($j,0), $centre($j,1))"
      $ns at [expr [$ns now]+ 0.022] "$SensorNode($tn) color red"
      $ns at [expr [$ns now]+ 0.022] "$SensorNode($tn) add-mark CC red circle"
      #The following line is to test the actual calculated center point
      #$ns at [expr [$ns now]+ 0.033] "$n($tn) setdest  $mean($j,0), $mean($j,1), 30"

      $SensorAgent($tn) set IsClusterHead "true"
   }
}

} ;# end of proc



for {set i 0} {$i < $num_nodes} {incr i} {
   set Location($i,0) 0
   set Location($i,1) 0
}


Agent/SensorAgent instproc recv {source sport size data} {
   $self instvar node_ SentBytes RecievedBytes IsClusterHead FusedDataSize
   global ns PercentageOfActiveDataSenders MaxFusedDataSize SensorDataSize
   set rng [new RNG]
   $rng seed 0
```

```
  if {[$node_ node-addr]==0 } {
    set RecievedBytes [expr $RecievedBytes + $MaxFusedDataSize]
  }

  if { $IsClusterHead =="true"} {
    if {$FusedDataSize < $MaxFusedDataSize } {
      set FusedDataSize  [expr $FusedDataSize + $size ]
    }
    if {$FusedDataSize >= $MaxFusedDataSize } {
      set FusedDataSize  0
      $ns at [$ns now] "$self SendSensorData $MaxFusedDataSize 0 $data"
    }
  }

}


Agent/SensorAgent instproc SendSensorData {size addr data} {
  $self instvar node_
  global ns SENSOR_AGENT_PORT

  $self sendto $size $data $addr $SENSOR_AGENT_PORT

}


#the following proceedure is to send Sensor Data Periodically
Agent/SensorAgent instproc SheduleSensorDataTransmission {size} {
$self instvar node_ ClusterHead SentBytes
global ns SensorDataInterval

  set now [$ns now ]
  #$ns trace-annotate "$now : [$node_ node-addr] is sending Data"
  #puts "$now : [$node_ node-addr]  is sending Data"

  set Data "DummySensorData"

  $ns at $now "$self SendSensorData $size $ClusterHead $Data"
  set SentBytes [expr $SentBytes + $size]
set rng [new RNG]
$rng seed 0
```

P-1815

```
   #Re-Schedule The Procedure
   set rr [$rng integer 10]
   set BroadcastJitter [expr 0.99/ ($rr.0+1) ]
   #puts $BroadcastJitter
   $ns at [expr $now + $SensorDataInterval + $BroadcastJitter ] "$self
SheduleSensorDataTransmission $size"
}

#Create Sensor Nodes
for {set i 0} {$i < $num_nodes } {incr i} {
     set x [expr [ $rng integer ($val(x)-150)] +125 ]
     set y [expr [ $rng integer ($val(x)-50)] +25 ]
     set SensorNode($i) [$ns node]
     #[lindex [Node($ni) array get netif_] 1] set Pt_ 0.1  ;# to limite the Transmission
Range around/below 100m

     $SensorNode($i) set X_ $x
     $SensorNode($i) set Y_ $y
     $SensorNode($i) set Z_ 0.0
     $SensorNode($i) color black


     #$ns at 0.0 "$Node($i) setdest $x.0 $y.0 3000.0"

     set SensorAgent($i) [new Agent/SensorAgent]
     $SensorNode($i) attach  $SensorAgent($i) $SENSOR_AGENT_PORT
     $SensorAgent($i) set IsClusterHead "false"
     $SensorAgent($i) set ClusterHead 0
     $SensorAgent($i) set SentBytes 0
     $SensorAgent($i) set RecievedBytes 0
     $SensorAgent($i) set FusedDataSize 0

     if { [$rng integer 100]<$PercentageOfActiveDataSenders} {
        if { $i > 0 } {
           $ns at 0.1  "$SensorAgent($i) SheduleSensorDataTransmission
 $SensorDataSize "
        }
     }

}
```

```
$SensorNode(0) color red
$SensorNode(0) set X_ 20
$SensorNode(0) set Y_ [expr $val(y) / 2 +100 |

for {set i 0} {$i < $num_nodes } {incr i} {
    $ns initial_node_pos $SensorNode($i) 20
}

#$ns at 0.00001 "$SensorNode(0) setdest 20.0 20.0 30000.0"
$ns at 0.0 "$SensorNode(0) color red"
$ns at 0.0 "$SensorNode(0) label BS/Sink"



# now set up some events
# now set up some events
#$ns at 25.51 "$SensorAgent(1) SendSensorData $SensorDataSize 1
{BroadcastMessageI}  $SENSOR_AGENT_PORT"
#$ns at 25.51 "$SensorNode(1) color gold"

$ns at 0.5 "$SensorAgent(0) FindLocationUsingGPSInfo"
$ns at 0.5  "$SensorAgent(0) kMeanCluster 7"

$ns at $TotalSimulationTime.1 "finish"


proc finish {} {
  global ns f nf val NamAnimationSpeed  num  nodes  SensorAgent
TotalSimulationTime
      $ns flush-trace
      close $f
      close $nf
      exec nam -r $NamAnimationSpeed Algorithm2.nam &
      exec ns graph.tcl &
      set TotalSentBytes 0.0
      set TotalRecievedBytes 0.0
      for {set i 0} {$i < $num_nodes } {incr i} {
        set  TotalSentBytes   [expr $TotalSentBytes + | $SensorAgent($i) set
SentBytes] ]

      }
```

```
    set  TotalRecievedBytes   [$SensorAgent(0) set RecievedBytes ]

    puts "Total Sent Bytes : $TotalSentBytes "
    puts "Total Recieved Bytes : $TotalRecievedBytes "
    puts "The Delivery Ratio : [expr $TotalRecievedBytes /$TotalSentBytes *100.0]"

    exec sh rtload Algorithm2.tr &
    exec sleep 2
    exec ns graph.tcl &

    #set Throughput  [expr
$TotalRecievedBytes.0/$TotalSimulationTime.0*8.0/1000000.0]
    #puts "The Average Throughput : $Throughput"

    exit 0
}


$ns run
```

7.6 Trace File

s -t 0.100000000 -Hs 0 -Hd -2 -Ni 0 -Nx 20.00 -Ny 350.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 0 -Iv 32
r -t 0.100000000 -Hs 0 -Hd -2 -Ni 0 -Nx 20.00 -Ny 350.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 0 -Iv 32
s -t 0.100000000 -Hs 1 -Hd -2 -Ni 1 -Nx 201.00 -Ny 33.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 1 -Iv 32
r -t 0.100000000 -Hs 1 -Hd -2 -Ni 1 -Nx 201.00 -Ny 33.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 1 -Iv 32
s -t 0.100000000 -Hs 2 -Hd -2 -Ni 2 -Nx 300.00 -Ny 272.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 2.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 2 -Iv 32
r -t 0.100000000 -Hs 2 -Hd -2 -Ni 2 -Nx 300.00 -Ny 272.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 2.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 2 -Iv 32
s -t 0.100000000 -Hs 4 -Hd -2 -Ni 4 -Nx 275.00 -Ny 489.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 4.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 3 -Iv 32
r -t 0.100000000 -Hs 4 -Hd -2 -Ni 4 -Nx 275.00 -Ny 489.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 4.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 3 -Iv 32
s -t 0.100000000 -Hs 5 -Hd -2 -Ni 5 -Nx 162.00 -Ny 157.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 5.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 4 -Iv 32
r -t 0.100000000 -Hs 5 -Hd -2 -Ni 5 -Nx 162.00 -Ny 157.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 5.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 4 -Iv 32
s -t 0.100000000 -Hs 6 -Hd -2 -Ni 6 -Nx 582.00 -Ny 397.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 6.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 5 -Iv 32
r -t 0.100000000 -Hs 6 -Hd -2 -Ni 6 -Nx 582.00 -Ny 397.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 6.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 5 -Iv 32
s -t 0.100000000 -Hs 7 -Hd -2 -Ni 7 -Nx 666.00 -Ny 299.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 7.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 6 -Iv 32

r -t 0.100000000 -Hs 7 -Hd -2 -Ni 7 -Nx 666.00 -Ny 299.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 7.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 6 -Iv 32
s -t 0.100000000 -Hs 8 -Hd -2 -Ni 8 -Nx 539.00 -Ny 304.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 8.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 7 -Iv 32
r -t 0.100000000 -Hs 8 -Hd -2 -Ni 8 -Nx 539.00 -Ny 304.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 8.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 7 -Iv 32
s -t 0.100000000 -Hs 9 -Hd -2 -Ni 9 -Nx 650.00 -Ny 134.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 9.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 8 -Iv 32
r -t 0.100000000 -Hs 9 -Hd -2 -Ni 9 -Nx 650.00 -Ny 134.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 9.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 8 -Iv 32
s -t 0.100000000 -Hs 10 -Hd -2 -Ni 10 -Nx 615.00 -Ny 289.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 10.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 9 -Iv 32
r -t 0.100000000 -Hs 10 -Hd -2 -Ni 10 -Nx 615.00 -Ny 289.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 10.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 9 -Iv 32
s -t 0.100000000 -Hs 12 -Hd -2 -Ni 12 -Nx 201.00 -Ny 674.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 12.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 10 -Iv 32
r -t 0.100000000 -Hs 12 -Hd -2 -Ni 12 -Nx 201.00 -Ny 674.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 12.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 10 -Iv 32
s -t 0.100000000 -Hs 13 -Hd -2 -Ni 13 -Nx 362.00 -Ny 304.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 13.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 11 -Iv 32
r -t 0.100000000 -Hs 13 -Hd -2 -Ni 13 -Nx 362.00 -Ny 304.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 13.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 11 -Iv 32
s -t 0.100000000 -Hs 14 -Hd -2 -Ni 14 -Nx 627.00 -Ny 647.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 14.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 12 -Iv 32
r -t 0.100000000 -Hs 14 -Hd -2 -Ni 14 -Nx 627.00 -Ny 647.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 14.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 12 -Iv 32
s -t 0.100000000 -Hs 15 -Hd -2 -Ni 15 -Nx 221.00 -Ny 365.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 15.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 13 -Iv 32

r -t 0.100000000 -Hs 15 -Hd -2 -Ni 15 -Nx 221.00 -Ny 365.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 15.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 13 -Iv 32
s -t 0.100000000 -Hs 16 -Hd -2 -Ni 16 -Nx 421.00 -Ny 490.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 16.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 14 -Iv 32
r -t 0.100000000 -Hs 16 -Hd -2 -Ni 16 -Nx 421.00 -Ny 490.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 16.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 14 -Iv 32
s -t 0.100000000 -Hs 17 -Hd -2 -Ni 17 -Nx 240.00 -Ny 198.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 17.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 15 -Iv 32
r -t 0.100000000 -Hs 17 -Hd -2 -Ni 17 -Nx 240.00 -Ny 198.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 17.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 15 -Iv 32
s -t 0.100000000 -Hs 20 -Hd -2 -Ni 20 -Nx 183.00 -Ny 93.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 20.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 16 -Iv 32
r -t 0.100000000 -Hs 20 -Hd -2 -Ni 20 -Nx 183.00 -Ny 93.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 20.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 16 -Iv 32
s -t 0.100000000 -Hs 21 -Hd -2 -Ni 21 -Nx 218.00 -Ny 544.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 21.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 17 -Iv 32
r -t 0.100000000 -Hs 21 -Hd -2 -Ni 21 -Nx 218.00 -Ny 544.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 21.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 17 -Iv 32
s -t 0.100000000 -Hs 22 -Hd -2 -Ni 22 -Nx 151.00 -Ny 582.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 22.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 18 -Iv 32
r -t 0.100000000 -Hs 22 -Hd -2 -Ni 22 -Nx 151.00 -Ny 582.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 22.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 18 -Iv 32
s -t 0.100000000 -Hs 23 -Hd -2 -Ni 23 -Nx 273.00 -Ny 325.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 23.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 19 -Iv 32
r -t 0.100000000 -Hs 23 -Hd -2 -Ni 23 -Nx 273.00 -Ny 325.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 23.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 19 -Iv 32
s -t 0.100000000 -Hs 25 -Hd -2 -Ni 25 -Nx 412.00 -Ny 671.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 25.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 20 -Iv 32

r -t 0.100000000 -Hs 25 -Hd -2 -Ni 25 -Nx 412.00 -Ny 671.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 25.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 20 -Iv 32
s -t 0.100000000 -Hs 27 -Hd -2 -Ni 27 -Nx 522.00 -Ny 269.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 27.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 21 -Iv 32
r -t 0.100000000 -Hs 27 -Hd -2 -Ni 27 -Nx 522.00 -Ny 269.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 27.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 21 -Iv 32
s -t 0.100000000 -Hs 29 -Hd -2 -Ni 29 -Nx 344.00 -Ny 243.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 29.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 22 -Iv 32
r -t 0.100000000 -Hs 29 -Hd -2 -Ni 29 -Nx 344.00 -Ny 243.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 29.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 22 -Iv 32
s -t 0.100000000 -Hs 30 -Hd -2 -Ni 30 -Nx 140.00 -Ny 162.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 30.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 23 -Iv 32
r -t 0.100000000 -Hs 30 -Hd -2 -Ni 30 -Nx 140.00 -Ny 162.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 30.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 23 -Iv 32
s -t 0.100000000 -Hs 31 -Hd -2 -Ni 31 -Nx 598.00 -Ny 228.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 31.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 24 -Iv 32
r -t 0.100000000 -Hs 31 -Hd -2 -Ni 31 -Nx 598.00 -Ny 228.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 31.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 24 -Iv 32
s -t 0.100000000 -Hs 32 -Hd -2 -Ni 32 -Nx 276.00 -Ny 468.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 32.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 25 -Iv 32
r -t 0.100000000 -Hs 32 -Hd -2 -Ni 32 -Nx 276.00 -Ny 468.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 32.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 25 -Iv 32
s -t 0.100000000 -Hs 33 -Hd -2 -Ni 33 -Nx 241.00 -Ny 199.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 33.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 26 -Iv 32
r -t 0.100000000 -Hs 33 -Hd -2 -Ni 33 -Nx 241.00 -Ny 199.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 33.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 26 -Iv 32
s -t 0.100000000 -Hs 34 -Hd -2 -Ni 34 -Nx 393.00 -Ny 445.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 34.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 27 -Iv 32

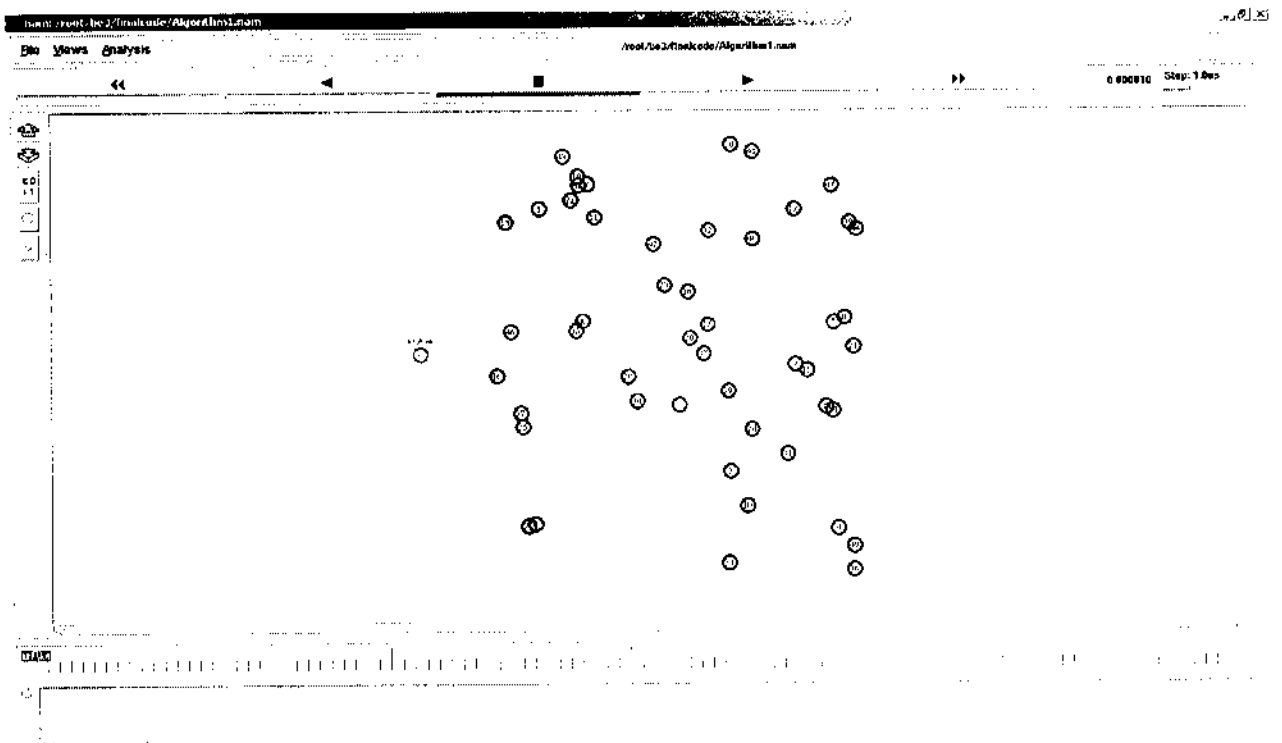
r -t 0.100000000 -Hs 34 -Hd -2 -Ni 34 -Nx 393.00 -Ny 445.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 34.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 27 -Iv 32
s -t 0.100000000 -Hs 35 -Hd -2 -Ni 35 -Nx 203.00 -Ny 217.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 35.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 28 -Iv 32
r -t 0.100000000 -Hs 35 -Hd -2 -Ni 35 -Nx 203.00 -Ny 217.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 35.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 28 -Iv 32
s -t 0.100000000 -Hs 38 -Hd -2 -Ni 38 -Nx 210.00 -Ny 623.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 38.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 29 -Iv 32
r -t 0.100000000 -Hs 38 -Hd -2 -Ni 38 -Nx 210.00 -Ny 623.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 38.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 29 -Iv 32
s -t 0.100000000 -Hs 39 -Hd -2 -Ni 39 -Nx 534.00 -Ny 613.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 39.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 30 -Iv 32
r -t 0.100000000 -Hs 39 -Hd -2 -Ni 39 -Nx 534.00 -Ny 613.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 39.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 30 -Iv 32
s -t 0.100000000 -Hs 41 -Hd -2 -Ni 41 -Nx 469.00 -Ny 367.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 41.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 31 -Iv 32
r -t 0.100000000 -Hs 41 -Hd -2 -Ni 41 -Nx 469.00 -Ny 367.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 41.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 31 -Iv 32
s -t 0.100000000 -Hs 42 -Hd -2 -Ni 42 -Nx 319.00 -Ny 306.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 42.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 32 -Iv 32
r -t 0.100000000 -Hs 42 -Hd -2 -Ni 42 -Nx 319.00 -Ny 306.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 42.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 32 -Iv 32
s -t 0.100000000 -Hs 43 -Hd -2 -Ni 43 -Nx 166.00 -Ny 191.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 43.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 33 -Iv 32
r -t 0.100000000 -Hs 43 -Hd -2 -Ni 43 -Nx 166.00 -Ny 191.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 43.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 33 -Iv 32
s -t 0.100000000 -Hs 44 -Hd -2 -Ni 44 -Nx 343.00 -Ny 404.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 44.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 34 -Iv 32

r -t 0.100000000 -Hs 44 -Hd -2 -Ni 44 -Nx 343.00 -Ny 404.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 44.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 34 -Iv 32
s -t 0.100000000 -Hs 45 -Hd -2 -Ni 45 -Nx 356.00 -Ny 280.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 45.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 35 -Iv 32
r -t 0.100000000 -Hs 45 -Hd -2 -Ni 45 -Nx 356.00 -Ny 280.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 45.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 35 -Iv 32
s -t 0.100000000 -Hs 46 -Hd -2 -Ni 46 -Nx 537.00 -Ny 428.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 46.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 36 -Iv 32
r -t 0.100000000 -Hs 46 -Hd -2 -Ni 46 -Nx 537.00 -Ny 428.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 46.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 36 -Iv 32
s -t 0.100000000 -Hs 48 -Hd -2 -Ni 48 -Nx 389.00 -Ny 166.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 48.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 37 -Iv 32
r -t 0.100000000 -Hs 48 -Hd -2 -Ni 48 -Nx 389.00 -Ny 166.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 48.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 37 -Iv 32
s -t 0.100000000 -Hs 49 -Hd -2 -Ni 49 -Nx 207.00 -Ny 392.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 49.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 38 -Iv 32
r -t 0.100000000 -Hs 49 -Hd -2 -Ni 49 -Nx 207.00 -Ny 392.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 49.42 -Id 0.42 -It message -Il 64 -If 0 -Ii 38 -Iv 32
s -t 0.100000000 -Hs 1 -Hd -2 -Ni 1 -Nx 201.00 -Ny 33.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.255 -Id -1.255 -It AODV -Il 48 -If 0 -Ii 0 -Iv 30 -P aodv -Pt 0x2 -Ph 1 -Pb 1 -Pd 0 -Pds 0 -Ps 1 -Pss 4 -Pc REQUEST
s -t 0.100000000 -Hs 2 -Hd -2 -Ni 2 -Nx 300.00 -Ny 272.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 2.255 -Id -1.255 -It AODV -Il 48 -If 0 -Ii 0 -Iv 30 -P aodv -Pt 0x2 -Ph 1 -Pb 1 -Pd 0 -Pds 0 -Ps 2 -Pss 4 -Pc REQUEST
s -t 0.100000000 -Hs 4 -Hd -2 -Ni 4 -Nx 275.00 -Ny 489.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 4.255 -Id -1.255 -It AODV -Il 48 -If 0 -Ii 0 -Iv 30 -P aodv -Pt 0x2 -Ph 1 -Pb 1 -Pd 0 -Pds 0 -Ps 4 -Pss 4 -Pc REQUEST
s -t 0.100000000 -Hs 5 -Hd -2 -Ni 5 -Nx 162.00 -Ny 157.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 5.255 -Id -1.255 -It AODV -Il 48 -If 0 -Ii 0 -Iv 30 -P aodv -Pt 0x2 -Ph 1 -Pb 1 -Pd 0 -Pds 0 -Ps 5 -Pss 4 -Pc REQUEST
s -t 0.100000000 -Hs 6 -Hd -2 -Ni 6 -Nx 582.00 -Ny 397.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 6.255 -Id -1.255 -It AODV -Il 48 -If 0 -Ii 0 -Iv 30 -P aodv -Pt 0x2 -Ph 1 -Pb 1 -Pd 0 -Pds 0 -Ps 6 -Pss 4 -Pc

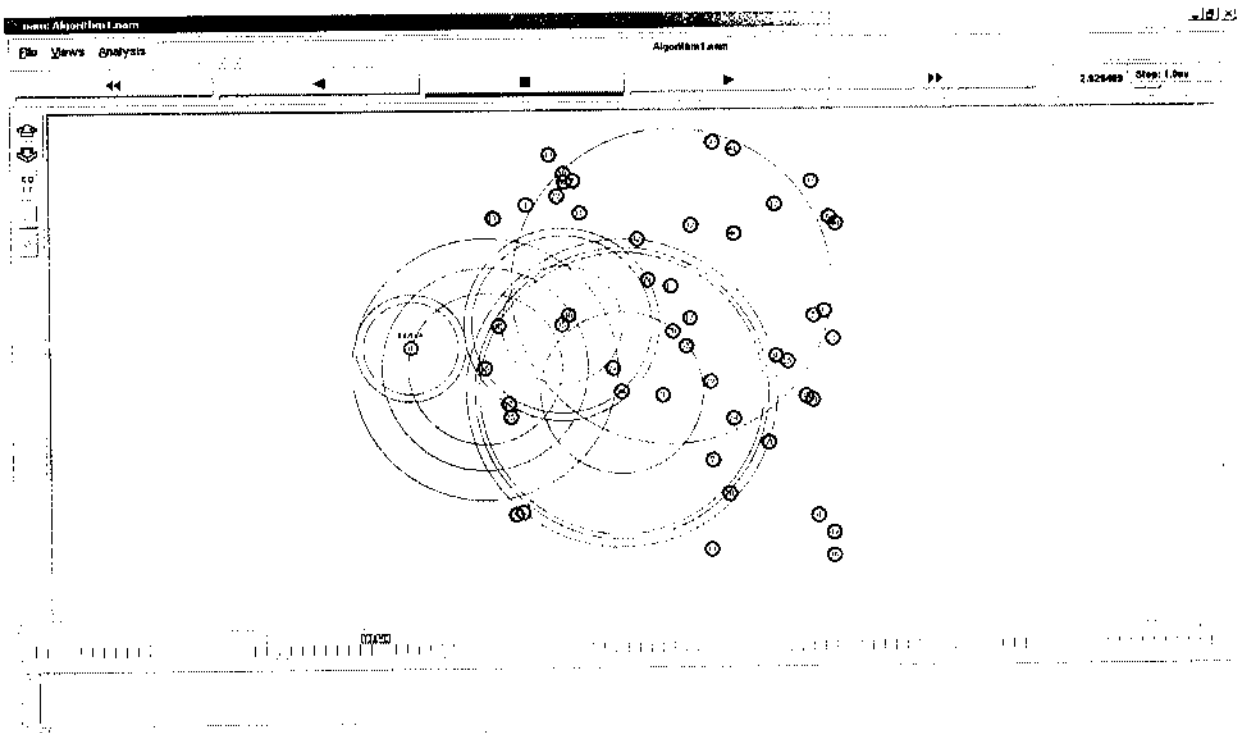
7.7 Network Animator Snapshot

Algorithm I – Normal Sensor Network

The 50 Node Sensor Network

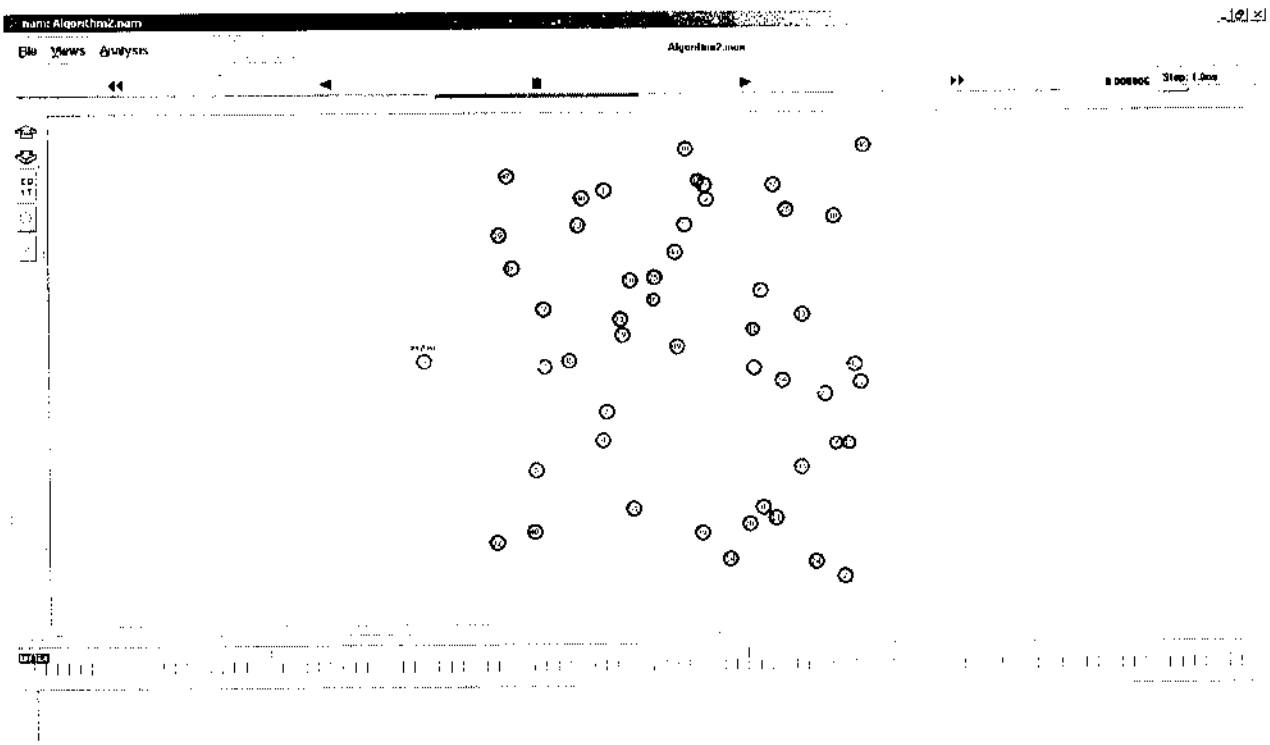


All the Sensor Nodes sending Data to the Sink at the periodic intervals

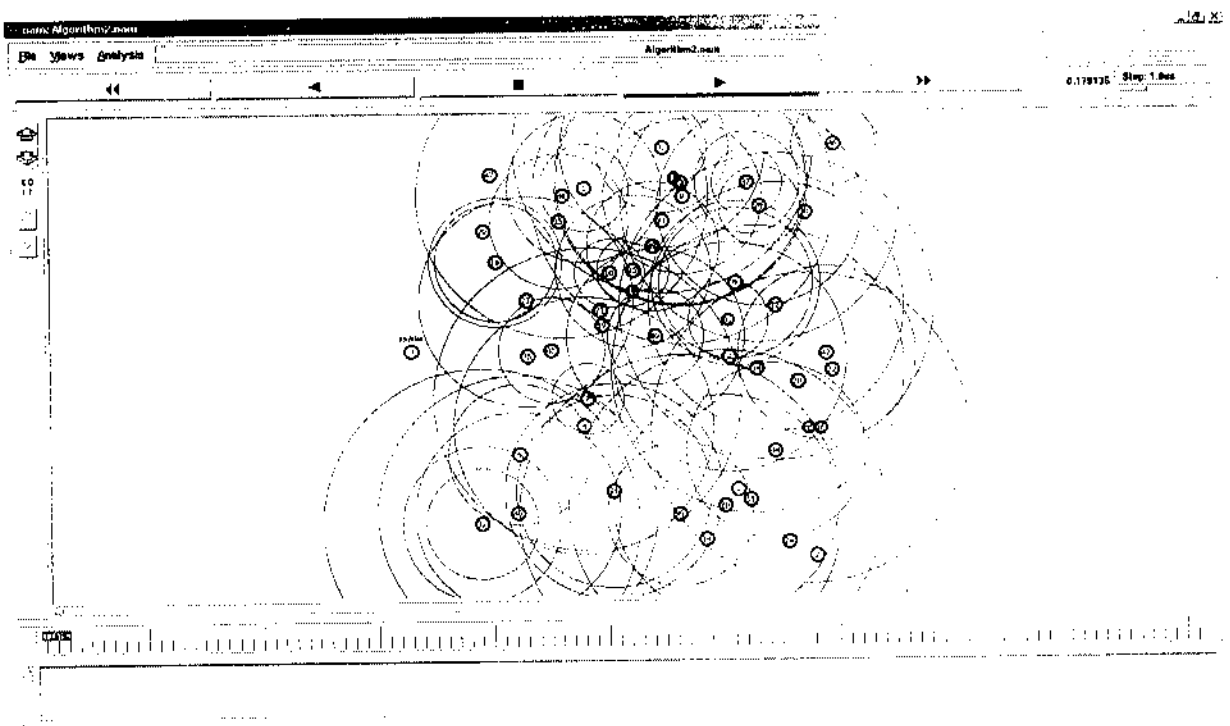


Algorithm II – Proposed Clustering Based Sensor Network

The 50 Node Sensor Network Before Clustering

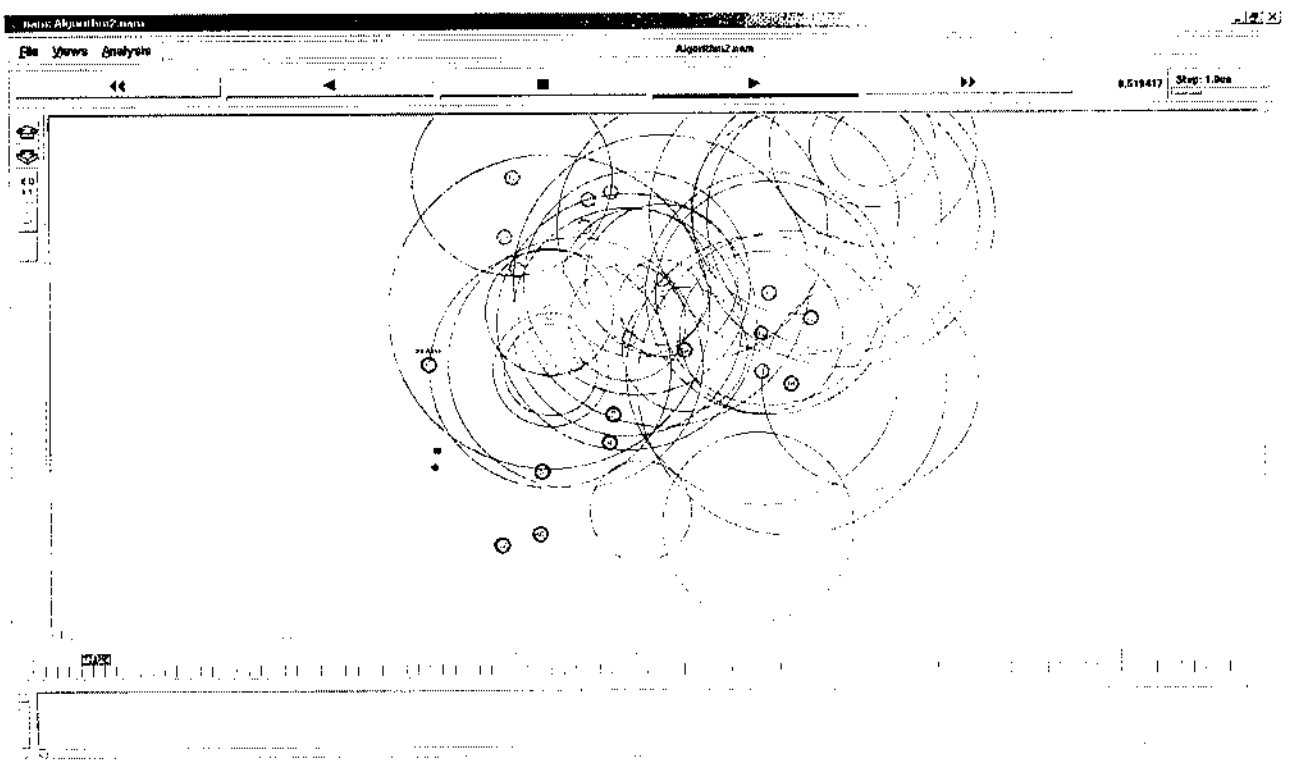


The Initial Message Broadcast for Route Discovery



The Clustered Sensor Network

- Each Cluster Member is shown in different colour.
- The Sink node (Left-Middle) is shown as red circle

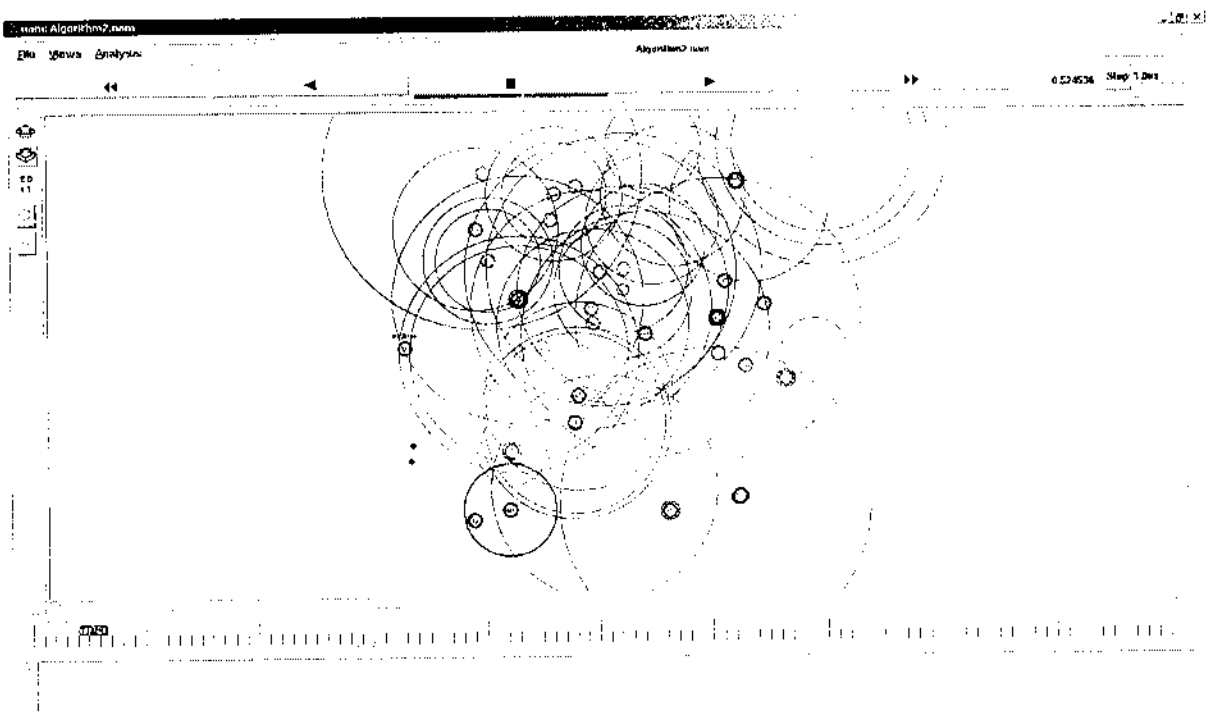


Marking Clusters and Cluster Centres ..
The Calculated Cluster Centres
215 174
413 92
237 471
513 383
567 103
492 599
646 282

The Sensor Network with 7 Selected Cluster Heads

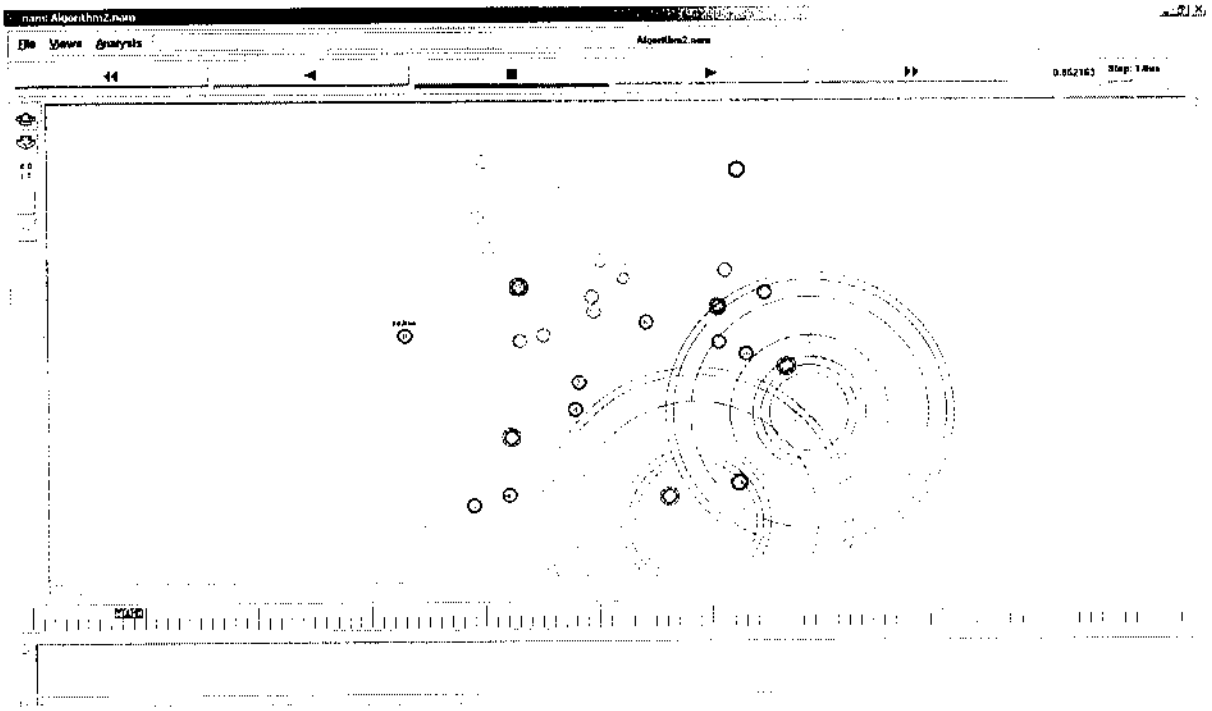
- The Cluster Heads are shown as marked red Circles (double lines)
- The Sink node (Left-Middle) is shown as red circle (single line)

The node Locations nearby the centers

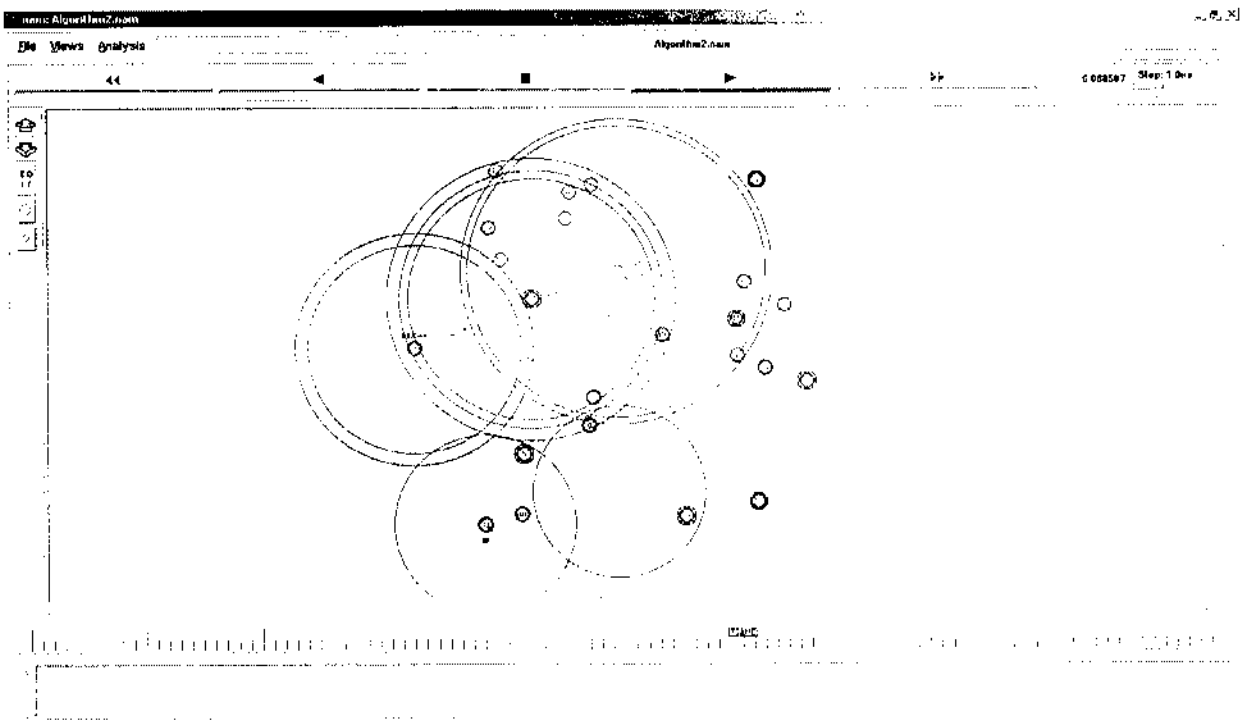


Node 5 : (186, 188)
Node 39 : (434, 93)
Node 27 : (198, 428)
Node 41 : (509, 396)
Node 11 : (543, 114)
Node 37 : (540, 614)
Node 28 : (617, 299)

The Sensor Heads are collecting Data Locally



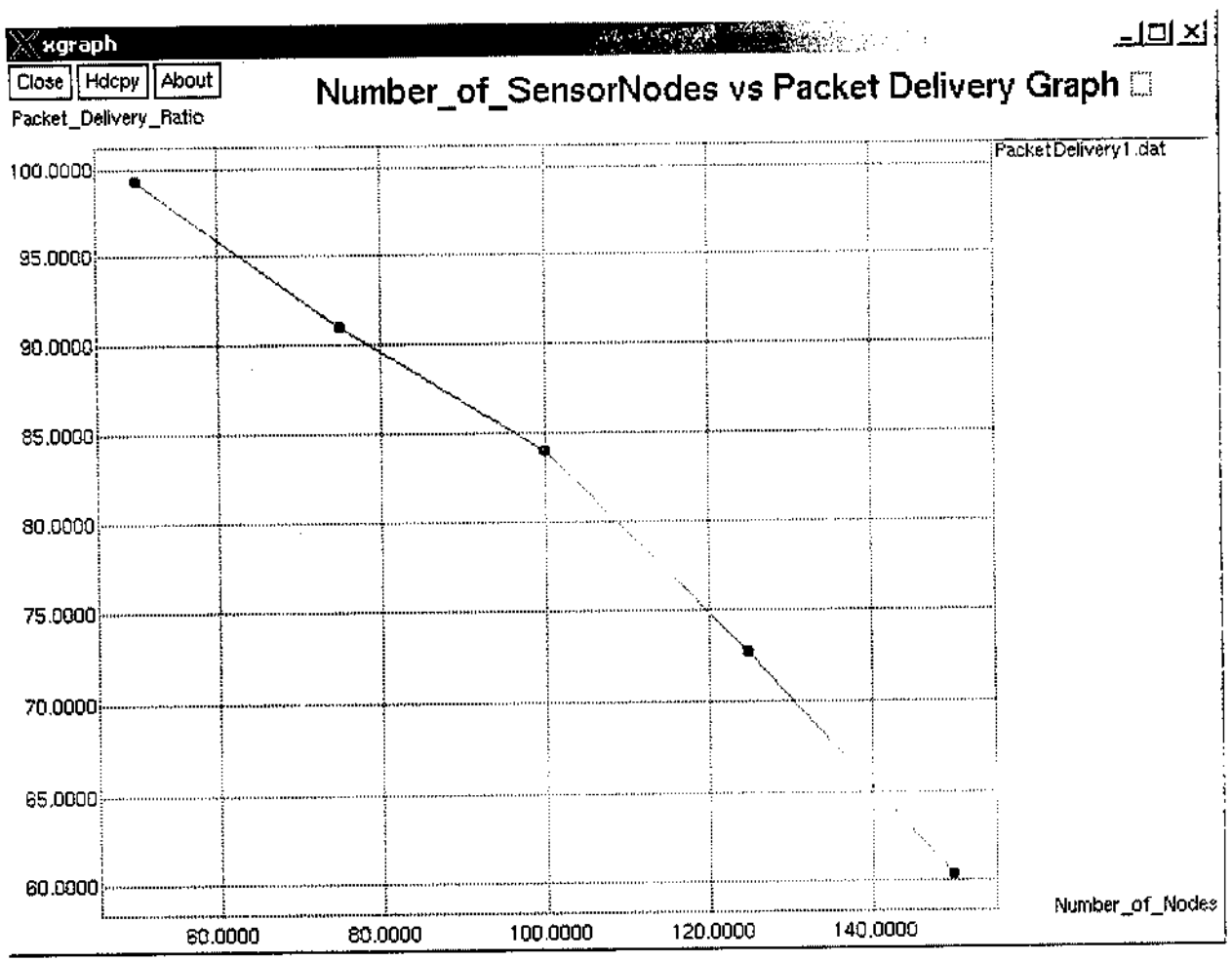
The Cluster Head is forwarding the collected Data to Sink



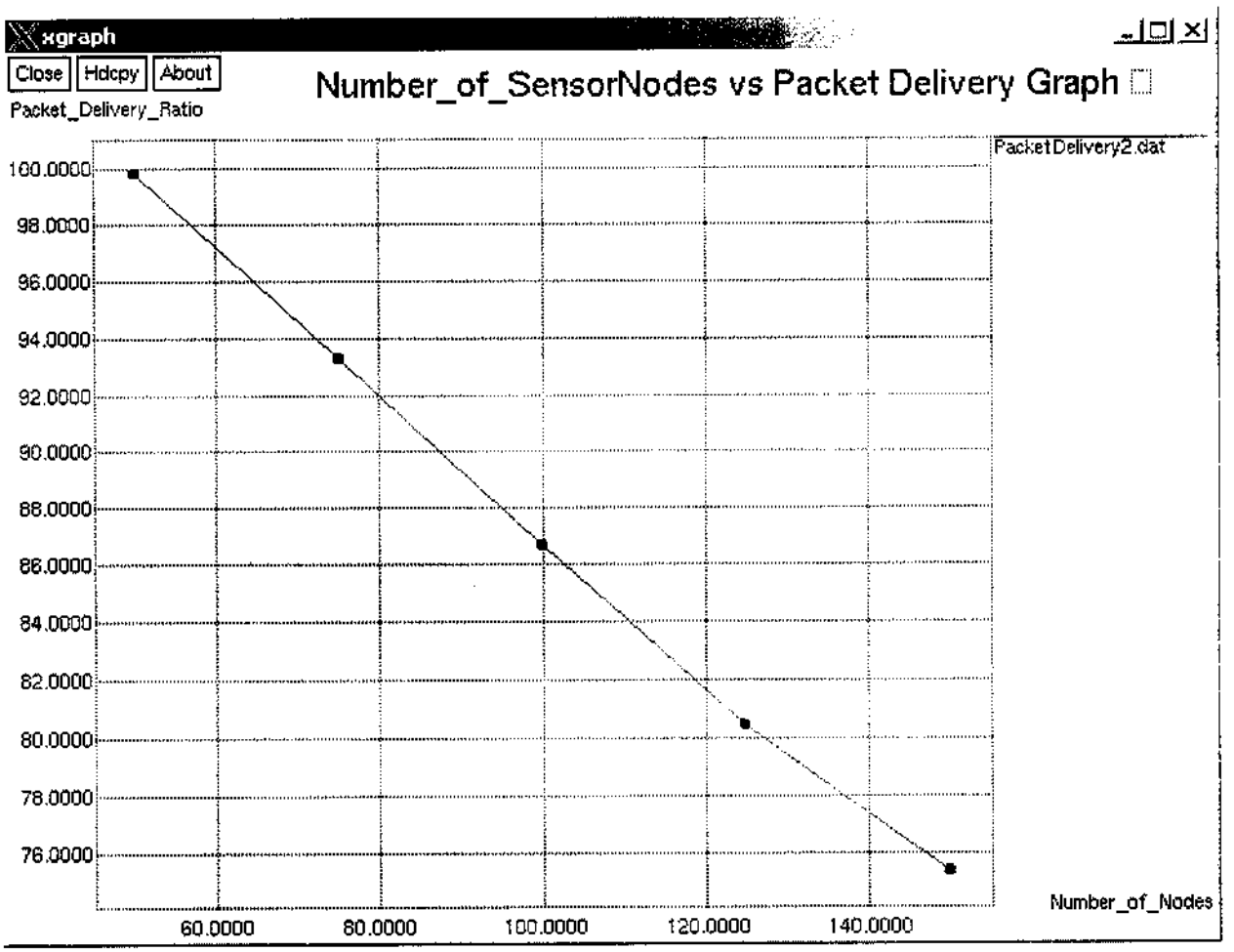
The Final Terminal Window output
Total Sent Bytes : 21056.0
Total Recieved Bytes : 20928
The Delivery Ratio : 99.392097264437695

7.8 Xgraph

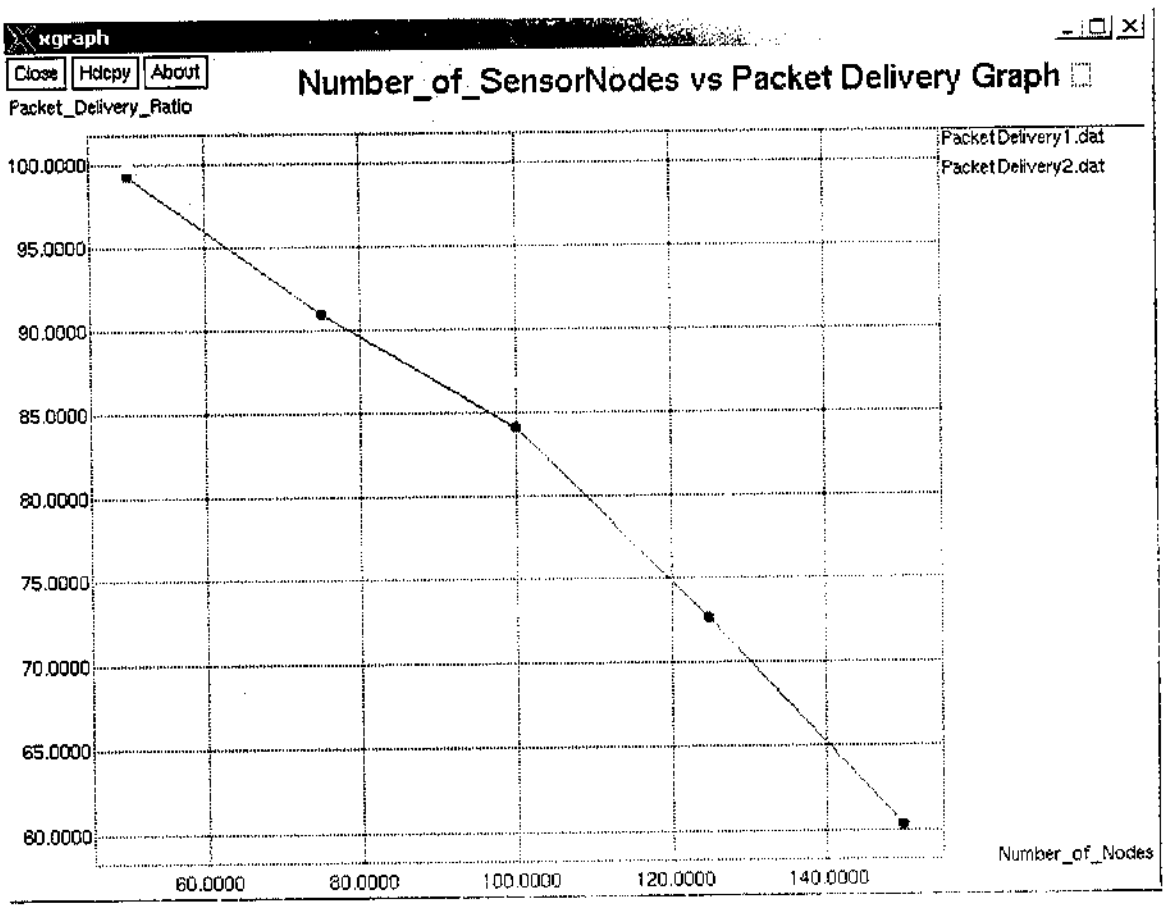
The Packet Delivery Ratio Graph(Without Clustering)



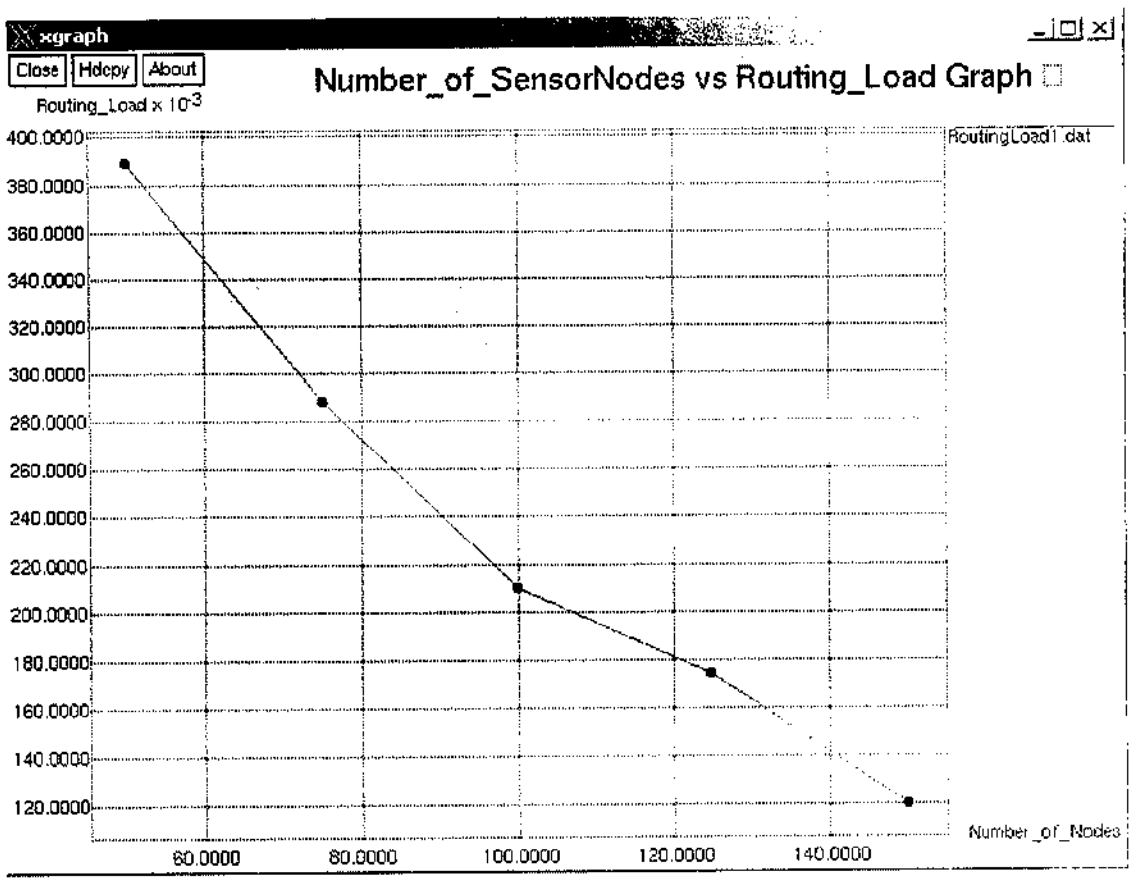
The Packet Delivery Ratio Graph(With Clustering)



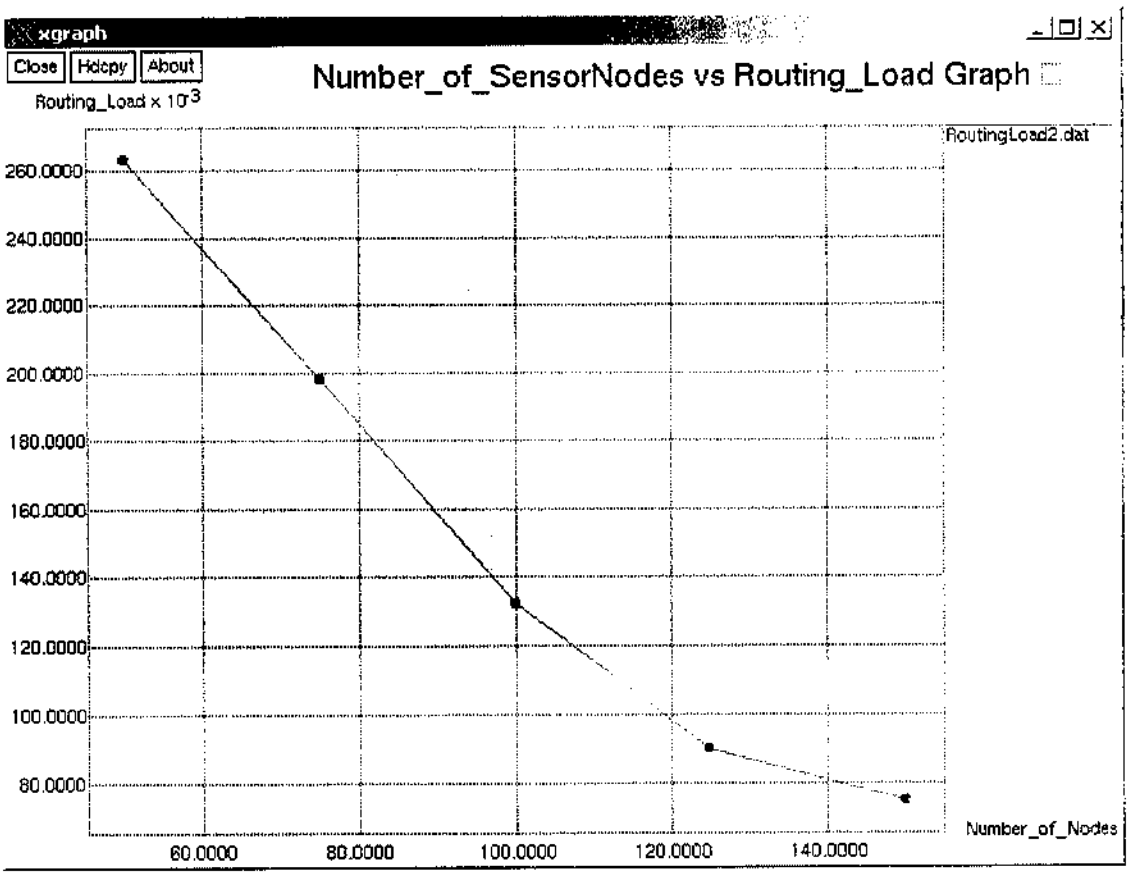
The Combined Packet Delivery Ratio Graph



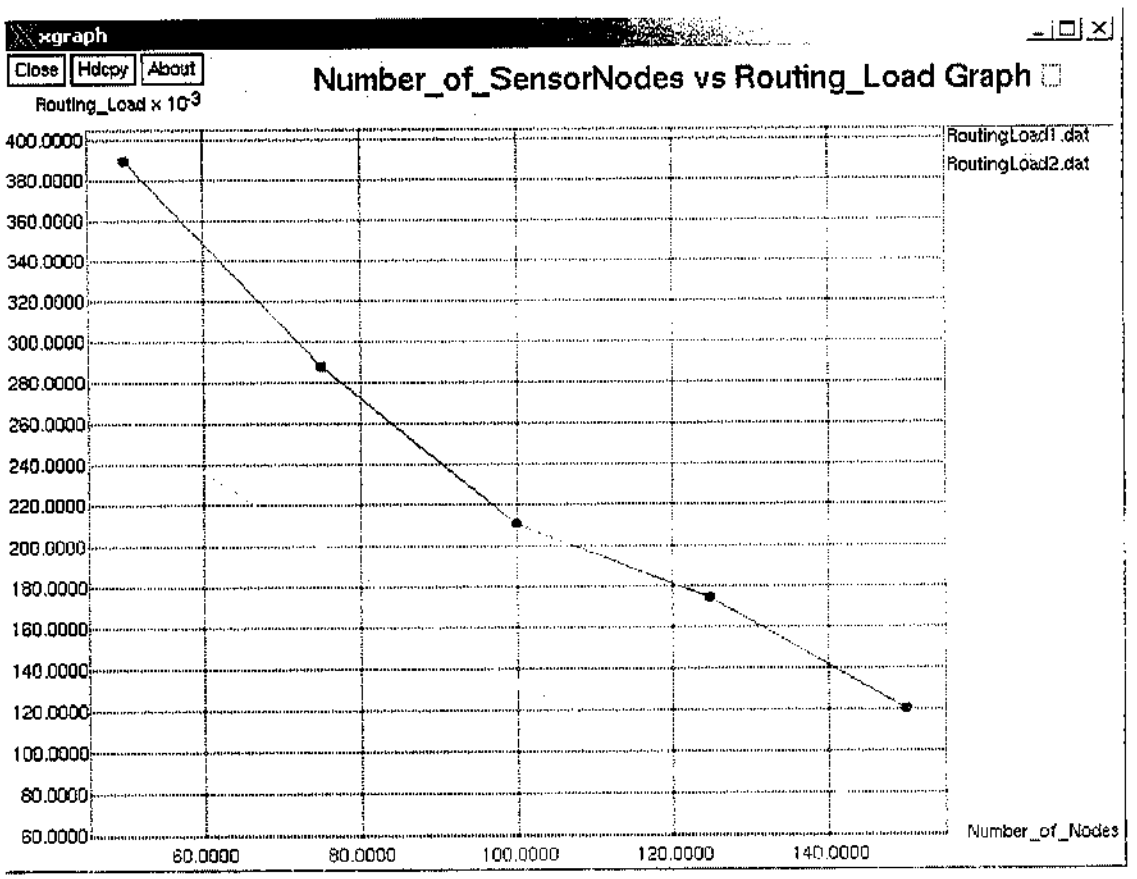
Number of SensorNodes vs Routing Load Graph (Without Clustering)



Number of SensorNodes vs Routing Load Graph (With Clustering)



Number of SensorNodes vs Routing Load Graph (Combined Graph)



7.9 Comparison Table

Table7.1: Number of SensorNodes vs Packet Delivery Ratio

Sl No	Number of Nodes	Packet Delivery Ratio	
		Normal Method	Clustering Based Method
1	50	99.25	99.80
2	75	90.87	93.23
3	100	83.98	86.66
4	125	72.67	80.45
5	150	60.32	75.34

Table 7.2: Number of SensorNodes vs Routing Load

Sl No	Number of Nodes	Routing Load	
		Normal Method	Clustering Based Method
1	50	0.389	0.263
2	75	0.288	0.198
3	100	0.210	0.132
4	125	0.174	0.090
5	150	0.120	0.075

7.10 Conclusion and Future work

Thus a comparison is done on the performance of routing methods for the traditional and the proposed hierarchical ad-hoc networks using k-means clustering. It is observed that the self-organizing hierarchical ad-hoc network performs well, and generally results in signicant improvements in both system capacity and end-user performance measures (such as delay and packet delivery fraction). Of course, the hierarchical architecture does require additional investment in forwarding node and access point equipment.Preliminary results are presented here from work in progress, and expected to address several open architectural and design optimization issues in future work. These topics include evaluation of alternative classes of ad-hoc routing (such as AODV) in the hierarchical scenario, design of more customized hierarchical routing protocols, discovery and routing protocols, and more complete system studies of realistic sensor network scenarios. It is also planned to implement selected hierarchical ad-hoc network protocols on the WINLAB sensor network testbed for the purposes of protocol validation and integration with real-world sensor applications.

8 REFERENCES

Papers:

[1] Yao-Chung Chang, Zhi-Sheng Lin and Jiann-Liang Chen (2006) '**Cluster Based Self-Organization Management Protocols for Wireless Sensor Networks**' Computer Science and Information Engineering Department, National Dong Hwa University, Hualien, Taiwan.

[2] John Lach1, David Evans2, Jon McCune3, Jason Brandon1 (2004) '**Power-Efficient Adaptable Wireless Sensor Networks**' Lingxuan Hu University of Virginia Departments of Electrical and Computer Engineering Pittsburgh.

[3] Boukerchey Ioannis Chatzigiannakis Sotiris Nikoletseaszi (2005) '**Power-Efficient Data Propagation Protocols for Wireless Sensor Networks' Aggregation**' School of Electrical and Computer Engineering,Unversity of Southern California.

[4] Seema Bandyopadhyay and Edward J. Coyle (2003) '**An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks**' School of Electrical and Computer Engineering, Purdue University.

[5] Vivek Mhatre, Catherine Rosenberg (2003) '**Design Guidelines for Wireless Sensor Networks: Communication, Clustering and Aggregation**' School of Electrical and Computer Engineering. Purdue University.

[6] Jamil Ibriq and Imad Mahgoub (2004) '**Cluster-Based Routing in Wireless Sensor Networks:Issues and Challenges**'Department of Computer Science,Florida Atlantic University

Website:

http://www.isi.edu/nsnam/ns/index.html

http://www.isi.edu/nsnam/ns/ns-documentation.html

http://tmml.sourceforge.net/doc/tcl/index.html

http://www.astro.princeton.edu/~rhl/Tcl-Tk_docs/tcl/contents.html

http://owfs.sourceforge.net/owtcl.n.html

Books:

[1] NS-2 manual

[2]Tutorial for ns simulator by Marc Greis.