DATA AGGREGATION IN SENSOR NETWORKS

by

SURYA TEJA KALLUMADI

B.Tech, Jawaharlal Nehru Technological University, 2005

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

Approved by:	Approved by:
Co-Major Professor Dr. Gurdip Singh	Co-Major Professor Dr. William H. Hsu

Abstract

Severe energy constraints and limited computing abilities of the nodes in a network present a major challenge in the design and deployment of a wireless sensor network. This thesis aims to present energy efficient algorithms for data fusion and information aggregation in a sensor network. The various methodologies of data fusion presented in this thesis intend to reduce the data traffic within a network by mapping the sensor network application task graph onto a sensor network topology. Partitioning of an application into sub-tasks that can be mapped onto the nodes of a sensor network offers opportunities to reduce the overall energy consumption of a sensor network. The first approach proposes a grid based coordinated incremental data fusion and routing with heterogeneous nodes of varied computational abilities. In this approach high performance nodes arranged in a mesh like structure spanning the network topology, are present amongst the resource constrained nodes. The sensor network protocol performance, measured in terms of hop-count is analysed for various grid sizes of the high performance nodes. To reduce network traffic and increase the energy efficiency in a randomly deployed sensor network, distributed clustering strategies which consider network density and structure similarity are applied on the network topology. The clustering methods aim to improve the energy efficiency of the sensor network by dividing the network into logical clusters and mapping the fusion points onto the clusters. Routing of network information is performed by inter-cluster and intra-cluster routing.

Index Terms

Sensor Networks, In-Network Processing, Data Fusion, Task Graph Mapping, Data Aggregation, Energy-Efficient Algorithms, Clustering

Table of Contents

List of Figures

Acknowledgements

I would like to thank my major professor, Dr. Gurdip Singh for his continued encouragement, invaluable guidance and ever-lasting patience. I owe my deepest gratitude to him for funding me during my Master's program. I am thankful to him for providing sound advice and lots of ideas during the course of my Master's.

I would also like to thank my co-major professor Dr. William H. Hsu for his support and willingness to point me to various Machine Learning concepts that have been utilised in this thesis. I am grateful to him for guiding me through the writing of this thesis and helping me by making requisite corrections and revisions to the text that is about to be read.

I have also to thank Dr. Doina Caragea for agreeing to serve on my Master's committee and for her willingness to review the thesis and suggesting numerous changes to the text of this thesis. I thank her for her time and effort.

CHAPTER 1 - Introduction

This chapter provides an overview of the thesis, a brief description of the basic concepts, the problem definition and motivation behind the thesis.

1.1. Motivation

Data transmission in large scale, densely distributed sensor networks presents many interesting and unique challenges. It is quite well known that data communication and message passing are one of the most expensive and energy intensive operations in wireless sensor networks [1]. Data aggregation and in-network processing techniques have been proposed as important mechanisms for routing in wireless sensor networks [2] [3]. The motivation behind data aggregation in sensor networks is to combine incoming data from diverse sources, within the network. This approach ensures that data redundancy is reduced if not completely eliminated, thus minimizing the number of messages and conserving scarce resources such as energy. Therefore, being able to transmit less data (the result of the aggregation over having to forward all the packets) results in reduced energy consumption at the sensor nodes. The data aggregation paradigm moves away from address-centric routing approach and focuses on a more efficient data-centric approach. Using in-network processing the computation work is pushed into the network, which performs aggregation before sending results to the base station. In this thesis we study the affect of data aggregation on the number of messages transmitted within the network, and show how network topology and network heuristics can help with efficient data aggregation strategies.

1.2. Introduction to wireless sensor networks

Rapid advances in micro sensors, wireless networking and embedded systems have enabled the development of distributed wireless sensor networks (WSNs). A WSN is a collection of autonomous computing nodes that systematically gather and transmit data in a distributed environment. In a sensor network the individual components interact in a distributed environment to achieve a common objective. A wireless sensor network application consists of a set of sensor nodes spread over a geographic region and the network collects information through these nodes. Sensor networks are usually dense networks and nodes in these networks share a common objective of data acquisition and information dissemination. Nodes are low cost miniature processing devices and each device or a node of the sensor network has the capability to sense an event and respond appropriately. The nodes acting as sensing devices have the ability to store data on local memory, process information attained and communicate appropriately with other nodes in the network. Nodes must cooperate with each other in a concerted manner to improve network efficiency and enhance the effective life time of the network. Apart from containing one or more sensors, these computing devices, commonly referred to as motes, are equipped with a radio transceiver. The radio transceiver inside a mote receives and transmits the data collected from one mote to another. Sensing motes present in the WSN sense and forward data packets to a root mote, also known as the sink. Motes in a sensor network can be regarded as tiny embedded computational devices with various constraints imposed upon them. Nodes in sensor networks are resource constrained with respect to battery lifetime or energy levels, processing speed and computational abilities, available memory, communication bandwidth and range. Because these motes in a WSN neither have a wired means of communication nor any human intervention involved in operating them, they have to be fully autonomous.

WSNs can be used in situations where data gathering and information processing is expensive and hazardous. They can be utilized in a wide variety of applications and systems.

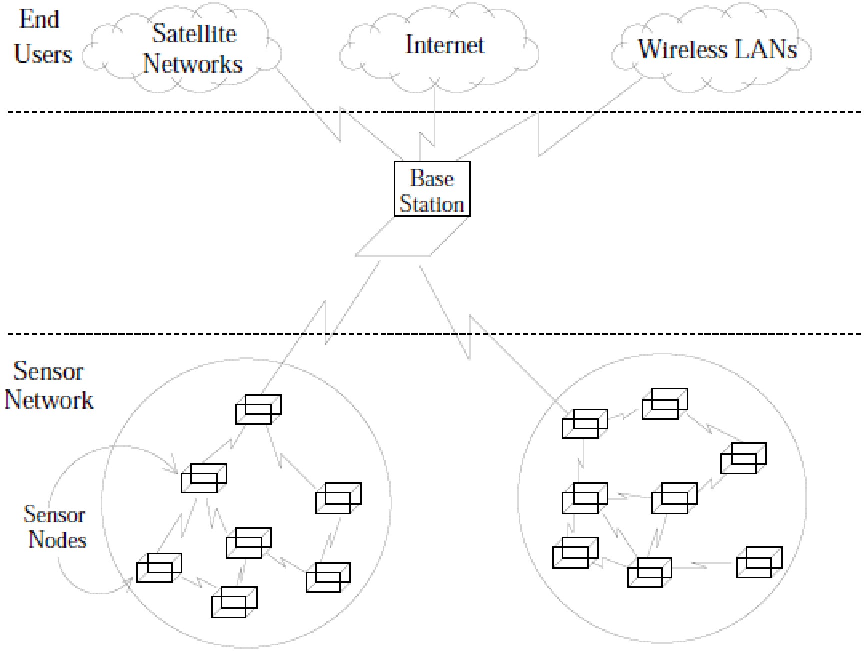


Figure 1.1 A typical WSN

Sensor networks can be used efficiently in varied fields such as object tracking [4], intrusion detection, environment monitoring [5] [6] etc. In addition, WSNs provide the technology for a wide range of systems in the military, thus creating new capabilities for intelligence gathering, reconnaissance and surveillance.

1.3. Routing strategies in a distributed sensor network environment

In a single sink, multi-source sensor network environment the sink node initiates the data gathering processing by disseminating interests within the network. In the interest dissemination phase the sink node intimates the network nodes about the kind of data it is interested in gathering. In the next phase the sensor nodes that match the interests respond to the sink by sending data to the sink. This data dissemination can be performed in multiple ways, and the way data is routed from the sources to the sink i.e., the routing strategy affects the efficiency and lifetime of a network. Routing strategies in sensor networks can be broadly classified into Address-centric routing and Data-centric routing [7].

Address-centric routing

In Address-centric routing each of the individual source nodes try to independently propagate data to the sink node on the shortest possible route. This end-to-end routing strategy introduces overhead and excessive message transmissions into the network, introducing network latency and reduced network life time.

Data-centric routing

In Data-centric routing individual source nodes attempt to transmit data to the source node by routing data through some common nodes in such a way that the intermediate nodes can perform data aggregation, thus introducing efficiency and consequently increasing the network lifetime.

1.4. Problem definition and objective

In this section we define the problem of data aggregation, the requirements and salient features of an ideal solution and how the output of role mapping should look.

Problem definition

Once a WSN has been deployed, a critical issue that needs to be addressed is efficient networked data gathering and information processing. This deals with the extraction and dissemination of sensor data from the network. Sensor nodes are severely handicapped by the limited amount of on board resources available to them. This includes energy, computing power and memory. The most critical resource in a WSN is the energy or the battery lifetime of a mote, making power conservation a critical aspect of WSN performance. Since the sensor nodes are energy-constrained, communication between the base station and the sensors must be energy-efficient. The key challenge in such an environment is the design of communication protocols that maximize the network lifetime. Network Lifetime is the time at which the first sensor node in the network dies i.e. it completely exhausts its battery resources. One of the generic ways of gathering data from a WSN is to directly send periodically extracted raw data from the sensor nodes to the sink. This, however, is a highly energy intensive exercise and thus a network life curtailing approach to information processing in a WSN. On the other hand a task graph based approach to gathering information from a sensor network can improve the energy efficiency and the effective life time of the network by reducing the amount of data that is being transferred in the network. Hence an appropriate way to describe the information which is to be retrieved from a sensor network is by using task graph. This method allows the sensor network application to be represented in the form of a task graph and is designed to describe the manner in which data has to be gathered and how it must be further transformed. Given a task graph and a network topology our objective is to map the task graph onto the topology in order to optimize a certain parameter or set of parameters.

Data fusion or aggregation is a useful paradigm in sensor networks. The key idea is to combine data from different sensors to eliminate redundant transmissions, thereby leading to efficient use of the energy resources.

Output format

We have a task graph mapped onto the network topology. A node in the network can be mapped to multiple nodes in the task graph.

Objective

Study how role assignment affects the scalability, reliability, robustness and responsiveness of the network. We also see how role assignment affects the life time of the network.

Solution requirements

The solution must be distributed and task mapping must be performed with limited knowledge of topology. As the conditions in the network change continuously, mapping must also be dynamic *i.e.*, mapping is a continuous process and solution should take these factors into account.

Proposed approaches for solution

This thesis will present an iterative, distributed role mapping algorithm that incrementally assigns intermediate computation steps onto fusion nodes within the network .When the topology of the network is known, network heuristics can be used for data aggregation in the network. We also look at how the nature of the task graph and construction of a grid or Backbone affects data aggregation in a structured sensor network topology. When the topology is not known we construct a logical topology by implementing distributed density-based clustering algorithms for data aggregation in sensor networks. Our proposed clustering solution simplifies data-centric routing in a dense network by adopting an intra-cluster and inter-cluster data-centric routing strategy for efficient data dissemination.

1.5. Contributions

My contributions to through thesis are

- Provided various approaches to role mapping in a sensor network environment.
- Explained the link between topology, task graph and the size of the grid Backbone and network efficiency
- Implemented density-based clustering in a distributed sensor network environment.
- Showed how density-based clustering can an efficient technique to improve network efficiency.

CHAPTER 2 - The role assignment problem

2.1. Task graph definition

We define a task graph as a directed acyclic graph which logically represents the data flow in a network. Task graph can be visualised as a tree of logical functions. The leaves of the tree correspond to the source nodes or sensing nodes that generate data. The internal nodes correspond to the fusion nodes that accept data from multiple sources and fusion nodes perform operations over the input data and generate data for consumption of other nodes higher in the task graph hierarchy. The root of the task graph corresponds to the Sink and gateway node. The edges represent the relationship that exists between two atomic execution units or tasks. There is a data dependency between two execution units. The nodes or points in the task graph are the execution points (that consume data from a previous node) of the application. These logical nodes have to be mapped onto the sensor network topology. Data fusion nodes in the task graph can have additional parameters (and, or, aggregation, max, min, fusion principle etc).In a TinyOS application each logical unit can be considered as a component and flow of data from one node to another can be considered as variables or messages that are relayed between the components.

2.2. Fusion operators

Fusion operators are used to represent operations on a set of atomic values. These fusion operators could be simple arithmetic operations, aggregation operations or even logical operations.

The various kinds of fusion operations could be:

- SUM produces the sum of the incoming data streams.
- AVG produces the average of the incoming data streams.
- MIN produces the lowest value of the incoming data streams.
- MAX produces the highest value of the incoming data streams.
- COUNT produces the number of items in the incoming data streams.
- OR forwards a value if fusion condition is satisfied in any of the incoming data streams.
- AND forwards a value if fusion conditions are satisfied in all the incoming data streams.

2.3. Aggregation operators

Apart from various fusion operators nodes in task graph can also be associated with various aggregation operators. The idea behind aggregation is that in a WSN sensed values in an area are related and instead of forwarding multiple redundant values into the network, it would be more efficient to send data that would be representative of the local conditions in a region.

The edges between the nodes have weights associated with them. These weights can be a single tuple or an "n" tuple. The weights can represent

- Rate at which data flows
- Conditions in which data flows
- Data expansion or contraction ratio etc.

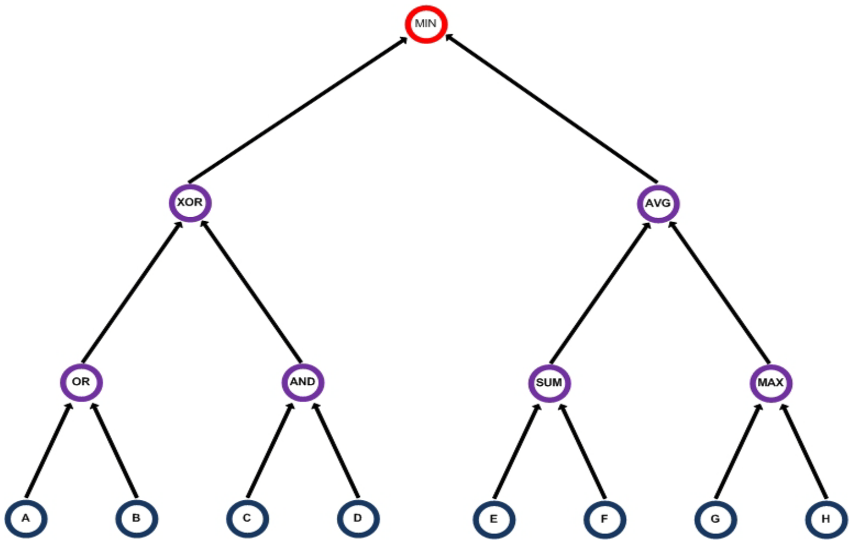


Figure 2.1 Illustration of a sample task graph

In Figure 2.1 Blue circles indicate the Source Nodes. Red circle is the Sink node and the purple circles are the various kinds of fusion nodes, Arrows indicate the flow of data.

Network Topology

A network topology consists of a set of sensor network nodes .Each node is associated with a geographical location (coordinates).The network nodes can be broadly classified as:

- Source node
- Sink nodes

- Aggregation nodes
- Data fusion nodes
- Bridge nodes/relay nodes

Source nodes are the nodes that generate data initially. Data could be either event triggered or time triggered. These are mapped to the leaf nodes in the Task graph.

Sink nodes are the end consumers of the data. These nodes are mapped to the Root node in the task graph. The sink nodes consume data generated by the task graph within the network.

Aggregation nodes are nodes that aggregate spatially related data. These nodes reduce data redundancy in the network and also generate data that represents a geographical area.

Data Fusion nodes are the nodes that fuse two or more data flow paths into a single path. Data fusion nodes are usually mapped to the internal computation nodes of the task graph that has 2 or more inbound edges (It can have multiple outbound edges).For a data fusion node in the task graph in-degree is greater than or equal to two.

Bridge nodes / relay nodes help transfer data between the above defined nodes.

Network topology can also be represented as a graph where the nodes represent the network nodes and the edges represent connectivity between the nodes. An edge indicates the presence of a single hop point to point connectivity between the nodes joined by the edge.

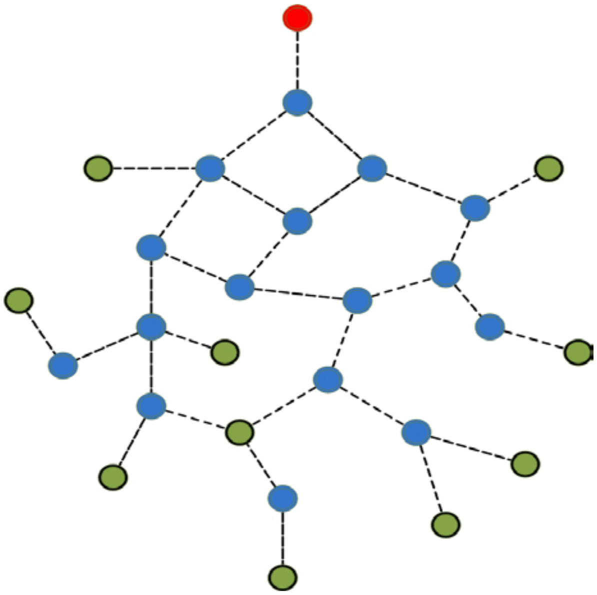


Figure 2.2 Sample WSN

The Figure 2.2 is an Illustration of sample a WSN, Green circles indicate the Sensing Nodes. Red circle is the Sink node and the Blue circles are the Bridge nodes, Dotted lines indicate the connectivity.

Nodes can have additional parameters such as

- Battery level
- Amount of free memory
- Average Length of task queue

2.4. The Role assignment problem in sensor networks

In this section we introduce the role assignment problem and provide a formal definition to the role assignment problem. We also show how role assignment problem is similar to the Steiner tree problem and why it is hard to find a solution to this optimization task.

A typical wireless sensor network topology consists of a large number of low-power wireless sensors spread across a geographical area that can be used to monitor and control the physical environment from remote locations and a centralized base station that receives data from various sensor nodes. The “base station” also known as a “sink node” gathers data from its source nodes and performs data processing tasks on the received information. As data communication and transmission cost is much more resource intensive and thus more expensive than data processing, the objective is to shift the data computation roles to within the wireless sensor network provided that such a role assignment reduces the amount of data transmitted. Task or role assignment involves mapping intermediate nodes of the application task graph onto the WSN topology to decrease the cost of data dissemination in the network. Role assignment drastically reduces the data traffic within the WSN and thus increases the lifetime of the network. Network lifetime is application specific and can be defined as duration for first node to fail or duration for at least one node to be active depending on the kind of application. Optimal placement of these intermediate fusion nodes constitutes the role assignment problem. Ideal placement of a fusion node i.e. task mapping would reduce amount of data transmitted within the network.

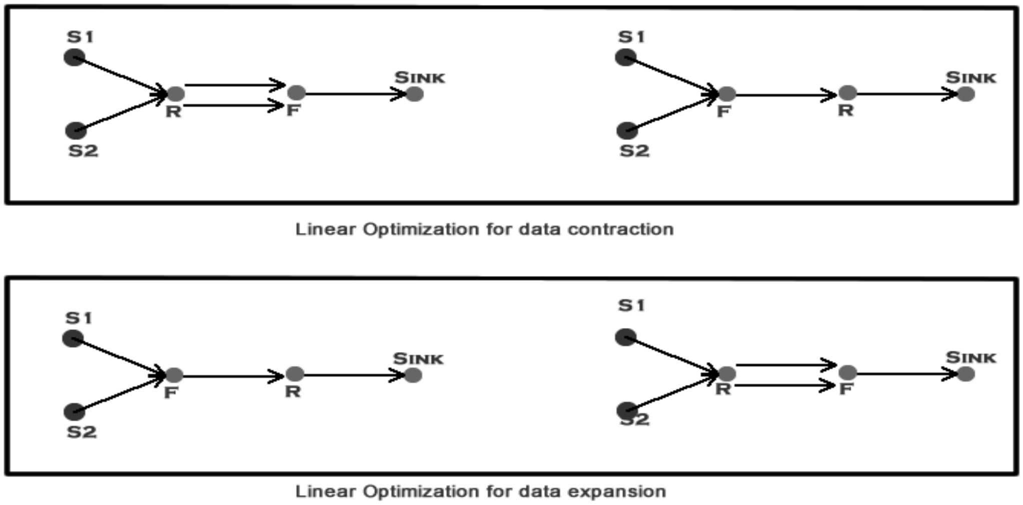
Types of data fusion

Placement of fusion node is a function of amount of data transmitted and path length of the tree.

Data fusion leading to data compression - In this case we try to map the fusion point as close as possible to the previous data source [Fig. 2.3].

Data fusion leading to data expansion- In this case we try to map the fusion point as close as possible to the consumer of the fused data [Fig. 2.3].

A Fusion application is a directed task graph and role assignment is similar to attaining a Minimum Steiner Tree [8] in a directed task graph. Role assignment is an NP-Complete problem.



Linear Optimization for data contraction

Linear Optimization for data expansion

Figure 2.3 Illustration of Linear optimisation

Steiner tree problem

Given a weighted graph G with a set of vertices V and weighted edges E and a set of terminals S such that S is a subset of V, the Steiner tree problem finds the minimum weighted connected sub graph that includes all the terminals S. The additional vertices apart from the terminal points S included to obtain the minimum Steiner tree are known as the Steiner points. The Steiner tree problem comes under the class of NP problems [9].There are two main aspects to the Steiner tree problem.

- Coming up with heuristics to determine what the Steiner points are.
- Choose Steiner points such that they contribute to a minimum cost.

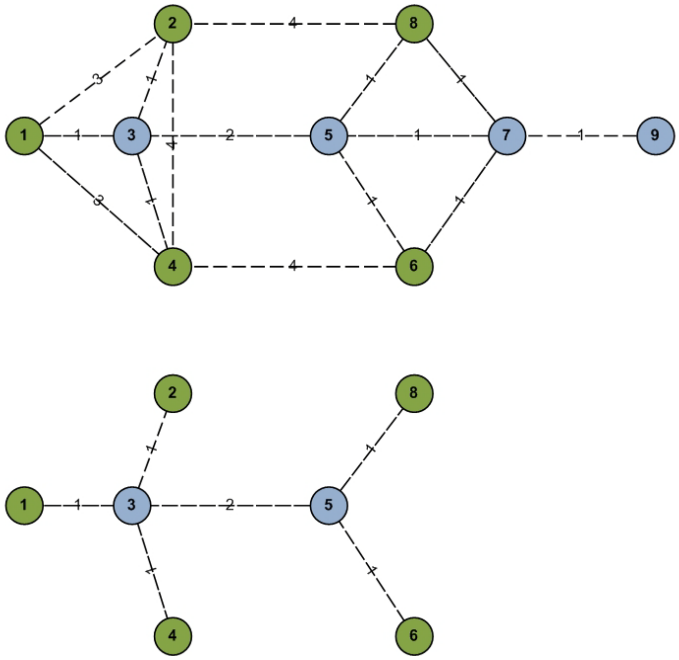


Figure 2.4 Illustration of Steiner tree mapping

The Figure 2.4 illustrates a graphical Steiner tree problem. The green nodes represent the Steiner points and the blue nodes represent the non-Steiner points. This is a graph theoretic approach to the Steiner tree problem. In the graph theoretic approach to the Steiner tree problem, let the terminal points to be connected be S and the additional vertices be set Z, both belonging to a larger set of vertices V in Graph G. These vertices are connected by edges whose length is given by the cost function C defined as

$$\mathbf{C: E \rightarrow R+}$$

The objective of minimal Steiner tree problem would be to minimize the total cost of the tree obtained from the graph G.

Formally represented as

$$\mathit{min}_{\mathbf{V^* \subset V\text{-}S}}\ \mathbf{C(T(S \cup V^*)}$$

Where the minimum cost is calculated over all the subsets $V* \subset V-S$ and T(S) and $T(V^*)$ represents the minimum cost spanning tree of G with vertices a S.

2.5. Role assignment problem and its resemblance to the Steiner tree problem

Mapping a task graph onto a sensor network is similar to a Steiner tree problem where the data sources in the task graph belong to the terminal set S and the nodes in the sensor network are the set of vertices V.

Task graph can be defined as a tree of functions. Each fusion node is associated with one or more functions, like aggregation, filtering, correlation etc.

Fusion node placement would be dependent on

- Location of its data generators
- In a direction towards the location of the next data consumer and the sink node
- Rate of data generation by the source nodes
- Distance between the data sources, sink and the next fusion node.

Optimal placement of the fusion node constitutes the task mapping problem.

Fusion node assignment will be closer to the node that has higher data rate in a direction towards the Sink node. It is also dependent on the data expansion/contraction ratios of the fusion function. A high data contraction ratio would tend to place the fusion node as close to the source nodes as possible closer to the source node that has the higher data rate.

The optimal placement of the fusion node depends on

- The rate at which sensor data is generated by the source nodes.
- The rate at which fusion node generates the fused data.
- The data expansion or contraction ratio of the fused and source data.
- Path length between the source nodes, fusion nodes and the Sink node.

Thus role assignment problem in WSN is to find the optimal assignment of fusion roles to the sensor nodes such that the mapping would minimize the amount of data transferred over the network. Apart from the above conditions in which role assignment takes place, the selection of a fusion node must be continuously and incrementally recomputed as the local conditions in the sensor network might change dynamically. It would be highly inefficient to have a static assignment of fusion nodes as the optimal placement of fusion node shifts as the data rates among the source nodes can change continuously and the available energy levels with the nodes might fluctuate. Thus the solution must also be adaptive and dynamic.

Role assignment can be accomplished under various conditions and assumptions. It could be assumed that the global topology of the WSN is known and this topological information can be used to map fusion roles onto the sensor network topology. Some amount of global knowledge of the WSN has to be maintained at every node in the scenario where the topological information is available to every node in the sensor network.

In other cases where a sensor network is deployed randomly, minimal amount of topological information could be available. This information would be mostly local with each individual node maintaining knowledge about nearby nodes. The solution to the role assignment problem in such conditions would have a purely distributed decentralized approach.

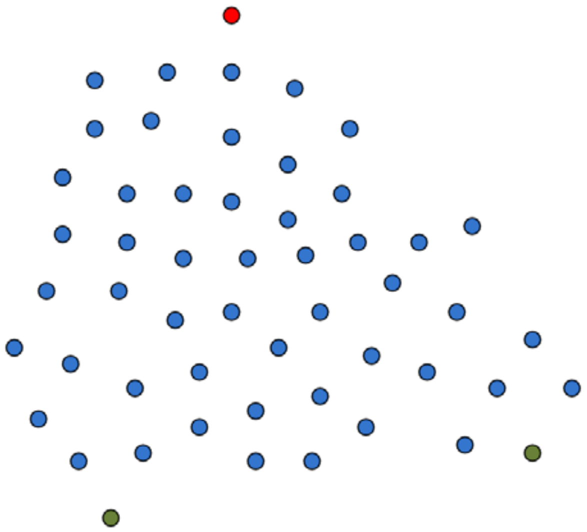


Figure 2.5 Illustration of a sample network topology

The above figure [Fig. 2.6] illustrates a sample wireless sensor network topology with nodes randomly distributed in a geographical area. The nodes within radio range can communicate with each other. The Green nodes are the source nodes which generate a continuous stream of data which might correspond to physical event occurring in the sensor network. The Red node is the Sink node which acts as a gateway between the sensor network and the outside world. The objective of the sensor network application is to detect events and aggregate them at the Sink node. The Blue nodes are the bridge nodes which act as the intermediate nodes used to route data between the Source nodes and the sink nodes.

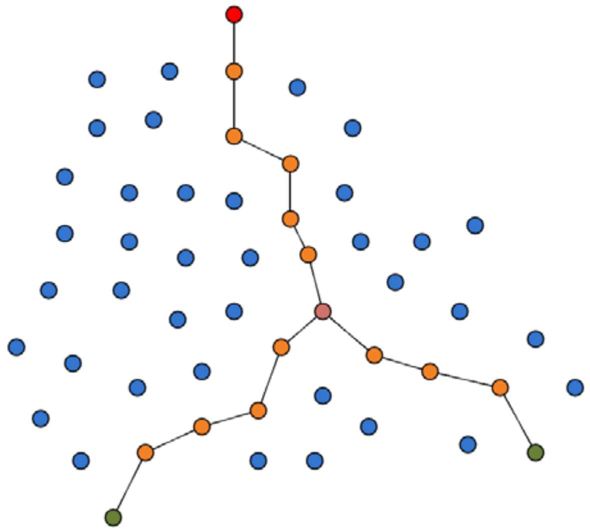


Figure 2.6 Mapping a task graph for uniform data rates

The above figure [Fig. 2.6] illustrates a case where there data contraction is happening in the sensor network and the rate at which data is generated by the source nodes is equal. The orange nodes are the Steiner points which are chosen in such a way that the total path cost is minimized. The Fusion point lies closer to the source nodes and in a direction toward the sink node. Because the data rates of the information generated by the source nodes are equal the fusion point tends to be equidistant from the Source nodes.

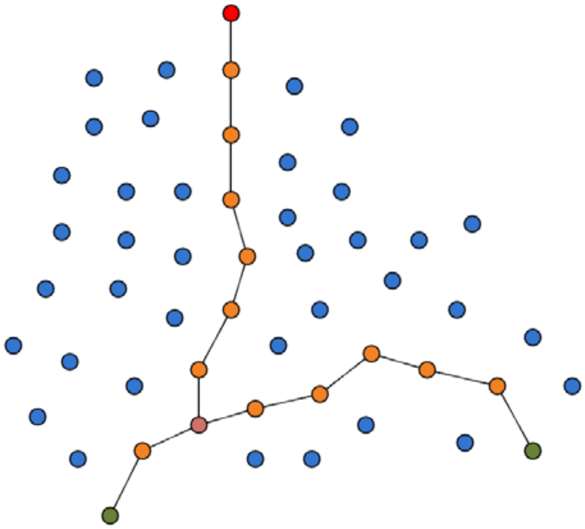


Figure 2.7 Mapping a task graph for non-uniform data rates

The above figure [Fig. 2.7] illustrates a case where the rate at which data is generated by the source nodes is unequal and there is data contraction in the sensor network. One of the source nodes produces information at a much higher rate than the other source nodes. The orange nodes are the Steiner points which are chosen in such a way that the total path cost is minimized. The Fusion point lies closer to the source node which produces more information and in a direction toward the sink node. As the data rates of the information generated by the source nodes are unequal the fusion point tends to be closer to the node that generates more data and further away from the node with less data rate.

Assumptions

Sensor nodes for this problem are assumed to be static and communicate with the base station in a multi-hop fashion. The sensor nodes periodically sense the environment and forward data in each round of communication. The end user can acquire data from a base station which is also a sensor node (the base station is usually referred to as a *sink node*). The intermediate nodes on the path to the sink, aggregate and fuse the data they receive from the others and forward the aggregate towards the sink. The problem is to find a routing mechanism that delivers data packets collected from sensor nodes to the base station in such a way, that it maximizes the lifetime of the sensor network.

Formal problem definition

We can define the task graph as a tree of Fusion operators. The leaves of the tree correspond to the source nodes that produce a steady stream of information. The internal nodes of the task graph are the various fusion operators like aggregation, logical, conditional operators etc. The root of the task graph is a fusion operator at which all the streams culminate. The root node is usually mapped to the sink/gateway node.

The aim of the task mapping problem is to map the fusion nodes such that it minimizes the amount of information transferred in the sensor network. A sensor network can be viewed as an undirected graph with the vertices playing the role of sensor node and the edges represent the communication links amongst these nodes.

A task graph is defined as

N - A set of nodes in the task graph.

L- The set of links connecting the nodes in the task graph.

WL – Weight associated with link L in the task graph, WL could be an n tuple. It denotes the rate at which data is transmitted between nodes n1 and n2 connected by link L.

A sensor network topology can be defined as:

S - A set of sensor nodes in the WSN topology.

C - The set of communication links connecting the nodes in S.

WC – Weight associated with the communication link C of the WSN topology S.

P(S1,S2) - The shortest path cost between two nodes S1 and S2 in the WSN S.

The Mapping of a task graph onto a sensor network topology can be defined as a

set M = {(n1,s1),(n2,s2),….} where the nodes $n \in N, s \in S$; the set of nodes n*={n1,n2,n3..} are defined as the Steiner points of the task mapping problem.

The Cost of sending data between node n1 and n2 of the task graph mapped onto nodes S1 and S2 in the WSN is defined as

$D_{n1,n2}(W_{L12}) = \sum P(S_a,S_b)$ where S_a,S_b are the nodes that are part of the path between S_1 and S_2.

The role mapping problem can be defined as the assignment of fusion nodes or the nodes of the task graph onto the WSN nodes such that it minimizes the global cost of communication given by : $\sum D_{n1,n2}(W_{L12})$

This role mapping problem is an instance of the minimal Steiner tree problem. The Steiner tree problem is known to be NP-complete. Although Steiner tree approximation algorithms are available, the algorithms are centralized and hence centralized Steiner tree approximation approaches are not suitable for a distributed sensor network.

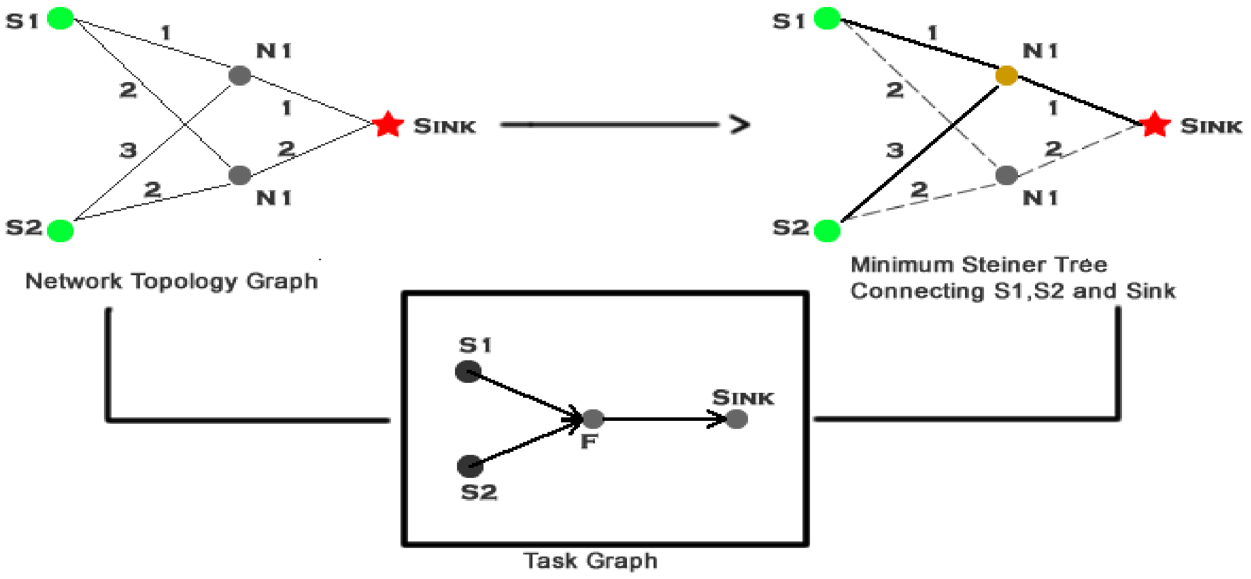


Figure 2.8 Mapping a Task Graph onto a Network

The above Figure 2.8 illustrates an instance of role mapping where a minimum Steiner tree is obtained by mapping the task graph onto the network topology.

CHAPTER 3 - Background and related work

3.1. Sensor hardware and deployment environment

In this chapter we will give a brief overview of some popular sensor network deployment and simulation environments.

An example hardware configuration of a sensor network node

The Berkley Mica Node [10] is a typical sensor network sensor/actuator mote with a CPU, power source, radio and several optional sensing elements. The processor is a 8-bit 4 MHz Atmel ATmega 128L processor with 128KB of program memory, 4KB of RAM for data and 512KB of flash memory. The processor only supports a minimal RISC-line instruction set, without support for multiplication or variable-length shifts or rotates. The radio is a 916 MHz low-power radio, delivering up to 40 Kbps bandwidth on a single shared channel and with a range of up to a few dozen meters. The radio consumes 4.8 mA in receive mode, up to 12 mA in transmit mode and 5 *micro* A in sleep mode.

TinyOS

TinyOS is an open source event driven light weight energy efficient sensor board operating system created by UCB [11]. The operating system handles the mote hardware, the radio controller for wireless communication and available memory by gather data from on board sensors, routing and sending messages to target nodes and controlling the energy consumption of the node. TinyOS uses a component oriented architecture approach to minimize code overhead and increase reusability. TinyOS' component-based architecture suits the application specific nature of WSNs. This also gives the system a certain level of flexibility for future designs. To conserve the available on board resources like battery power TinyOS incorporates an event based execution model so that the applications are event driven and the resources used for event are released once the event is handled. To optimize resource utilization the program components respond to events or hardware interrupts. Networked nodes in a WSN may not be physically accessible. Generally there is no end-to-end communication between the motes and the sink, and radio messages being transmitted may not be received. For these reasons WSNs adopt ad hoc network formation. TinyOS' multi-hop networking architecture takes care of this, as well as providing support for more heterogeneous networks.

nesC

nesC is programming language designed for low power sensor nodes with high resource constraints [12]. Because of the resource constraints on the mote hardware, nesC programming language is used to exploit the modular design and reusable code concepts of TinyOS. nesC has C like syntax and includes the code efficiency and simplicity of C language. nesC programs are built by wiring individual components through a set of interfaces that specify the commands it provides and the events it handle. Interfaces provide for bidirectional interaction between the component and its user.

3.2. Modeling and simulation environments

NS-2

NS-2 is a well-established open-source network simulator from UCB [13]. It is a discrete event simulator with substantial support for simulation of TCP, routing and multi-cast protocols over wired and wireless networks.

OPNET

OPNET Modeler developed at MIT offers sophisticated modeling and simulation of communication networks [14]. An OPNET model is hierarchical where the top level contains the communication nodes and the topology of the network. It uses a discrete-event simulator to execute the entire model.

Ptolemy-II

Ptolemy-II integrates diverse models of computation, such as continuous-time, discrete event, finite state machines, process networks, synchronous data flow, synchronous/reactive. This capability can be used, for example to model the physical dynamics of sensor nodes, their digital circuits, energy consumption, signal processing or real-time software behavior.

TOSSIM

TOSSIM is the TinyOS network Simulator. It can support a large number of motes and accurately model the timing interactions between motes [15]. For these reasons, TOSSIM is a suitable simulator for testing and evaluating WSNs. TOSSIM allows easy transition between simulation and real-world WSNs. The event queue supplies events which are essentially hardware interrupts. The arrival of each event allows TinyOS to continue running its applications. TOSSIM makes use of a simple, but effective abstraction to model WSNs. It considers a wireless sensor network to be a directed graph, each vertex representing a sensor node and each edge having a bit error probability associated with it. This enables users to model packet transmission failures or a perfect network with no transmission errors and failures, by adjusting the bit rate errors. The event-driven nature of TOSSIM, which goes hand in hand with that of a WSN, allows users to set breakpoints in what normally is a real-time simulation. TOSSIM allows other programs to interact with the simulation. TinyViz is one such program.

TinyViz

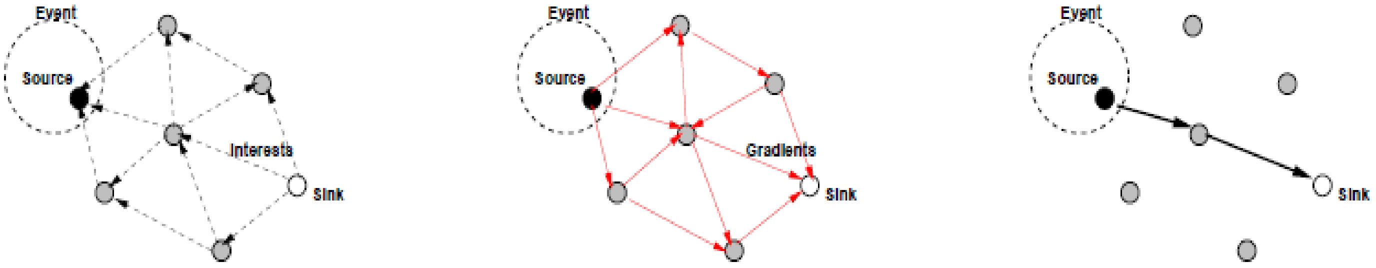
TinyViz, a visualization tool for TOSSIM, is a Java based Graphical User Interface (GUI). TinyViz is developed to aid the evaluation and debugging of WSNs by providing useful visualizations of sensor networks. A number of basic plug-ins are supplied by TinyViz, allowing users to monitor network traffic, examine debugging statements, set breakpoints which pause the simulation when certain events take place, and set the radio connectivity range of motes based on their relative distance on the display. TinyViz not only allows developers to extend the currently available plug-ins, but also enables the implementation of application-specific visualizations through the creation of entirely new plug-ins, which are run within the TinyViz engine. The interaction between TOSSIM and TinyViz, as well as its plug-in based architecture, makes TinyViz a suitable visualization tool for simulations.

3.3. Literature review and existing approaches

In this section we present some of the commonly used techniques for collecting data in a sensor network environment.

Directed Diffusion

Directed Diffusion is a data-centric communication paradigm for sensor networks [16]. In directed diffusion, data generated by the sensors is named by attribute-value pairs. A sensing task (initiated by the sink) is disseminated throughout the sensor network as an *interest* for the named data. This interest dissemination sets up *gradients* within the network that point to the neighbor from which an interest was received. Sensors matching the interest send their data to the sinks along multiple gradient paths initially, and then gradually reinforce better paths. Intermediate nodes aggregate the data and forward the fused data to the next node till it reaches the sink. Fig. 3.1 gives an illustration of directed diffusion.



(a) Interest propagation

(b) Initial gradients set up

(c) Data delivery along reinforced path

Figure 3.1 Directed Diffusion

SPIN - Sensor Protocols for Information via Negotiation

SPIN efficiently disseminates information among sensors in an energy-constrained wireless sensor network [17]. Nodes running SPIN name their data using high-level data descriptors, called metadata. SPIN nodes base their communication decisions both upon application-specific knowledge of the data and upon knowledge of the resources that are available to them. This allows the sensors to efficiently distribute data given a limited energy supply.

PEGASIS - Power-Efficient GAthering in Sensor Information Systems

In PEGASIS, the authors propose a new chain-based protocol called PEGASIS that minimizes the energy consumption at each sensor node [18]. PEGASIS achieves reduction in energy consumption as compared to LEACH since it requires only one designated node to send the combined data to the base station. The key idea is that nodes organize to form a chain and each node takes turns being the leader for communication to the base station. The data is collected starting from each endpoint of the chain and aggregated along the path to the designated head-node. Unlike LEACH (that uses hierarchical clustering), PEGASIS uses a flat topology thereby eliminating the overhead of dynamic cluster formation. PEGASIS achieves a better performance than LEACH by between 100% and 300% in terms of network lifetime.

PEDAP - Power Efficient Data Gathering and Aggregation Protocol

In PEDAP, the authors propose a new minimum spanning tree-based protocol called PEDAP and its power-aware version (PEDAPPA) [19]. The data packets are routed to the base station over the edges of the minimum spanning tree. PEDAP outperforms LEACH and PEGASIS by constructing minimum energy consuming routing for each round of communication. The advantage with PEDAP-PA is that it minimizes the total energy of the system while distributing the load evenly among the nodes. This leads to increased system lifetime.

APTEEN - Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network Protocol

APTEEN uses an enhanced TDMA schedule to efficiently incorporate query handling [20]. APTEEN provides a combination of proactive (by requiring nodes to periodically send data) and reactive (by making nodes to respond immediately to time-critical situations) policies.

3.4. Introduction to Clustering and Clustering in wireless sensor networks

In this section we provide a brief overview of clustering, explain some popular clustering approaches and methodologies and review the commonly used clustering approaches in sensor networks.

Clustering

The objective of clustering is to divide data into meaningful groups and through this process discover useful but not so obvious information present in large collection of data objects [21]. Clustering aims at grouping data such that objects within groups are similar while objects in different groups are dissimilar [Fig. 3.2]. The greater the similarity within the objects of a cluster, and the greater the difference between clusters, the better is the clustering technique. The better clustering technique tries to maximize intra-cluster similarity and minimize the inter-cluster similarity. Because clustering methods do not assume the presence of prior knowledge of data to be clustered, clustering is called as an unsupervised learning technique.

Based on the need for clustering and the application domain, cluster membership can be subject to multiple definitions.

A cluster can be defined as a grouping in which every member of the cluster is identical to every other member of the cluster and less similar to other cluster objects. A threshold is used as a similarity measure.

A cluster can also be defined as a set of objects in which all the members are similar to the representative member of the cluster commonly called as a centroid or center of gravity than with the centers of other clusters. The centroid could be a medoid in which case a cluster object acts as a cluster center or a centroid i.e. the average of all the members of the cluster.

Clusters can also be defined as regions of high-density separated by low-density regions. This approach to clustering is mostly used to discover clusters of arbitrary size and shape. This is popularly known as density based approach to clustering.

Data representation for cluster analysis

Data objects in cluster analysis are usually represented as vectors in an *n* dimensional space where each dimension represents an attribute or measurement that partially describes the data object. Thus the data objects to be clustered are represented as an *m-by-n* matrix where each of the m rows represent the individual data objects and the n columns represent the attributes of the data vector.

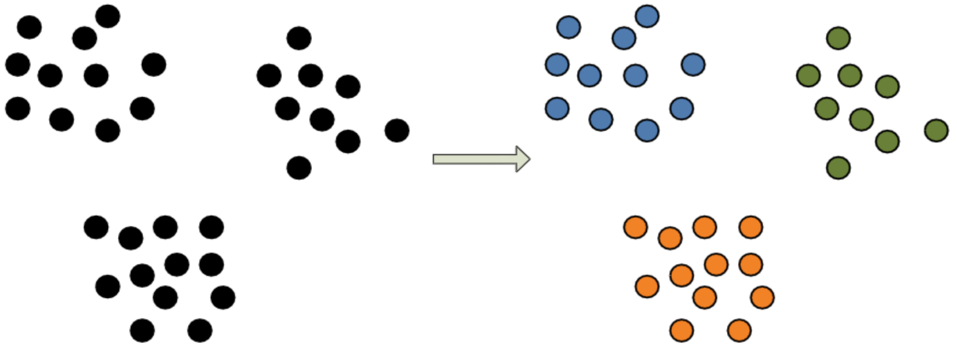


Figure 3.2 Clustering

Proximity measures

Minkowski metric or Distance Measures

Minkowski metric is the most popular and most commonly used distance measure to determine the proximity of two data objects [22]. This metric is a generalization of the normal distance between two points in a Euclidian space. It is defined as

$$P_{ij}=[\ \sum_{k=1 to\ d} |\ x_{ik}-x_{kj}\ |^{r}\ ]^{i/r}$$

Where r is the distance parameter, d is the dimensionality of the data object and x_{ik},x_{kj} are the respective k^th^ components of the i^th^ and j^th^ objects of x_i and x_j.

The following is a list of the common Minkowski distances for specific values of r.

r=1 Manhattan distance *aka* L1 distance

r=2 Euclidian distance aka L2 distance, usually used to calculate the distance between two points in the Euclidean space.

Jaccard's coefficient for binary vectors

Jaccard's coefficient is used to calculate the similarity between binary vectors [23]. Jaccard's similarity coefficient has values between 0 and 1. A value of 1 indicates that the two vectors are completely similar, while a value of 0 indicates that the vectors are not at all similar.

The comparison of two binary vectors, p and q, leads to four quantities:

M_{01} = the number of positions where p was 0 and q was 1

M_{10} = the number of positions where p was 1 and q was 0

M_{00} = the number of positions where p was 0 and q was 0

M_{11} = the number of positions where p was 1 and q was 1

The *Jaccard coefficient* measure is given by

J = (M11) / (M01 + M10 + M11)

Cosine similarity measure

The cosine measure, defined below, is the most common measure similarity between non-binary vectors. If *d1* and *d2* are two document vectors, then

cos(*d1, d2*) = (*d1* • *d2*) / ||*d1*|| ||*d2*|| is the cosine similarity measure

Where • indicates vector dot product and ||*d*|| is the length of vector *d*.

3.5. Traditional clustering approaches

Clustering methods are typically either based on distances (like partitioning and hierarchical clustering) or on densities (like density-based methods).

Partition techniques

One of the most common approaches to cluster a data set is clustering by partitioning. As the name suggests, clustering by partitioning creates a disjoint non-overlapping grouping of the data set. Partitioning algorithms iteratively improve an initial partition of the data until a cost function converges. Usually in partition algorithms the number of clusters into which the data has to be clustered has to be mentioned. If K is the desired number of clusters, then partition approaches typically find all K clusters at once.

Hierarchical techniques

Usually as the number of clusters are not known hierarchical techniques come up with a nested sequence of partitions represented in the form of a binary tree structure where a single universal cluster is at the root and individual data objects are at the leaf level [Fig. 3.3]. The intermediate levels can be viewed as clusters formed by some proximity metrics. There are two kinds of hierarchical techniques namely divisive (top-down) and agglomerative (bottom-up) clustering methods. Divisive methods start with a single cluster that contains all objects and recursively pick one cluster for splitting from the top. Starting with a single cluster and ending up with individual data items. Agglomerative methods assign each object to an individual cluster and then iteratively merge the two closest clusters together using a chosen distance function. By choosing different levels in the hierarchical tree we can obtain a different clustering of the data set.

Density-based techniques

Density based clustering techniques define clusters as dense regions separated by sparsely populated regions. Density of a region can be measured by either a simple count of the objects or by using complex models for density determination. Density based techniques are useful for detecting arbitrarily shaped clusters in noisy settings.

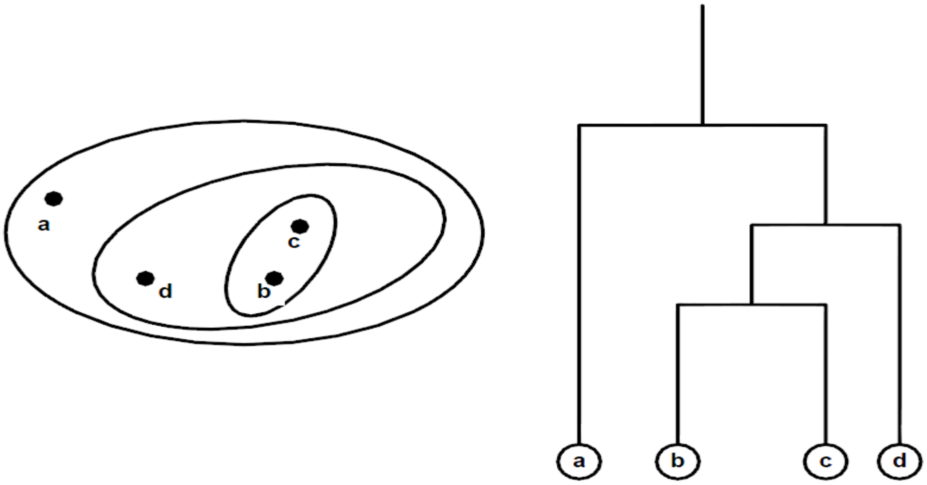


Figure 3.3 Hierarchical clustering

3.6. Partition techniques

K-Means

The *K-Means* algorithm is a partition based clustering technique. It is based on the idea that a center point of a cluster can represent the cluster .K-Means uses the concept of a centroid which is nothing but the center of gravity of the cluster which is the mean or median of all the data points associated with that particular cluster. The centroid in K-means may or may not be an actual data point [24].

K-means algorithm for finding K clusters

I. Select *K* points as the initial centroids.
II. Assign each point in the data set to the closest of the *K* centroids.
III. Recalculate the centroid of each cluster.
IV. Repeat steps 2 and 3 until the centroids don’t change.

Choosing the initial centroids is the critical step of the K-means clustering. In K-means algorithm the initial choice of the K centroids can determine the final outcome of the clustering. Depending on the initial choice of the centroids clustering solution can converge to either a global minimum or a local minimum.

Time and space complexity

As only the *n* dimensional data vectors are stored to represent each of data points, the space complexity of the K-means algorithm is *O(mn)*,where *m* is the number of points in the data set and *n* is the number of attributes or dimensions of each data point. The complexity of the K-means algorithm is *O(I*K*m*n),* where *I* is the number of iterations required for the K-means algorithm to converge. *I* is usually a small value and can be easily bounded as most changes occur in the first few iterations. Thus, K-means is linear in *m*, the number of points, and is efficient, as well as simple, as long as the number of clusters is significantly less than *m*.

Formal algorithm for K-means

Given the data set X, choose the number of clusters $1 < c < N$.

Initialize with random cluster centers chosen from the data set X.

Repeat for $l = 1, 2\ldots$

Step 1 Compute the distances

$$\mathbf{D^2ik = (x_k - v_i)^T (x_k - v_i); 1 \leq i \leq c; 1 \leq k \leq N}$$

Step 2 Select the points for a cluster with the minimal distances, they belong to that cluster.

Step 3 Calculate cluster centers

$$\mathbf{v}_i = \sum_{j=1 \text{ to } Ni} \mathbf{x}_i / \mathbf{N_i} \quad \text{until} \prod_{k=1 \text{ to } n} \mathit{max}|\mathbf{V^{(l)}-V^{(l-1)}}| \neq 0$$

K-Medoid or Partition around medoids (PAM)

The *K-medoids* algorithm is a partition based clustering technique similar to the K-Means technique [25]. The objective of *K-medoid* clustering is to find a non-overlapping set of clusters such that each cluster has a most representative point, i.e., a point that is most centrally located with respect to some measure, *e.g.,* distance. In *K-medoid* we use the concept of a medoid, which is the most representative (central) point of a group of points belonging to the cluster. By definition a medoid is required to be an actual Data point. Thus unlike K-Means, in *K-Medoid* the center point which represents the cluster must essentially be a data point.

Basic K-medoid algorithm for finding K clusters.

I. **Select *K* initial points.** These points are the candidate medoids and are intended to be the most central points of their clusters.

II. **Consider the effect of replacing one of the selected objects (medoids) with one of the non-selected objects.** Conceptually, this is done in the following way. The distance of each non-selected point from the closest candidate medoid is calculated, and this distance is summed over all points. This distance represents the "cost" of the current configuration. All possible swaps of a non-selected point for a selected one are considered, and the cost of each configuration is calculated.

III. **Select the configuration with the lowest cost.** If this is a new configuration, then repeat step 2.

IV. **Otherwise, associate each non-selected point with its closest selected point (medoid) and stop.**

Formal algorithm for K-medoids

Given the data set X, choose the number of clusters $1 < c < N$.

Initialize with random cluster centers chosen from the data set X.

Repeat for $l = 1, 2\ldots$

Step 1 Compute the distances

$$D^2ik = (x_k - v_i)^T (x_k - v_i);\ 1 \leq i \leq c;\ 1 \leq k \leq N$$

Step 2 Select the points for a cluster with the minimal distances, they belong to that cluster.

Step 3 Calculate k-means cluster centers

$$v_i = \sum_{j=1 \text{ to } Ni} x_i / N_i$$

Step 4 Choose the nearest data point to be the cluster center

$$D^2ik = (x_k - v^*_i)^T (x_k - v^*_i)$$

and

$$x_i^* = argmin_i\ D^2ik^* \Rightarrow V_i = x_i^*$$

Until $\prod_{k=1 \text{ to } n} max|V^{(l)} - V^{(l-1)}| \neq 0$

3.7. Distributed clustering techniques for sensor networks

Distributed clustering techniques are used in sensor networks to increase scalability of traditional protocols and reduce network delays. To support scalability the sensor nodes are grouped into non-overlapping disjoint clusters that interact with each other [**26**]. Clustering simplifies routing by reducing the size of routing table and divides the routing problem into intra-cluster and inter-cluster routing and in the process create a logical topology for a sensor network whose nodes are not location aware. The following are some clustering algorithms in a WSN environment.

LCA - Linked cluster algorithm

The focus of LCA is primarily on forming an efficient network topology that can handle the mobility of nodes [**27**]. In LCA clustering, cluster heads are joined to form a Backbone network to which cluster members can connect while on the move. The Objective of the proposed distributed algorithm is to form clusters such that a Cluster head is directly connected to all nodes in its cluster. LCA is thus geared for maximizing network connectivity. The algorithm assumes synchronized nodes and time-based medium access.

Adaptive clustering

Adaptive Clustering looks to optimally control the cluster size by balancing the interest in the spatial reuse of channels, which is increased by having small clusters, and data delivery delay, which gets reduced by avoiding inter-cluster routing, i.e. large cluster sizes[**28**]. In adaptive clustering every cluster would use a distinct code resulting is simplified implementation and great potential for meeting the Quality of Service requirements often found in multimedia applications.

CLUBS

CLUBS is an algorithm that forms clusters through local broadcast and converge in a time proportional to the local density of nodes [29]. Basically, cluster formation in CLUBS is based on network connectivity, cluster diameter and intra-cluster connectivity within clusters. The algorithm forms clusters with a maximum of two hops. Each node in the network takes part in the cluster formation process by choosing a random number from a fixed integer range. Then it counts down from that number silently. If the countdown was not interrupted from any other neighboring node and it reaches zero, it announces itself Cluster Head and broadcasts a ‘‘recruit’’ message. When a neighboring node receives the recruit message that comes within two-hop diameter boundary, it stops the countdown, accepts the invitation and joins the cluster. A node that has joined a cluster is called ‘‘follower’’ is no longer allowed to compete for being a Cluster Head.

Hierarchical control clustering

Objective of Hierarchical control clustering is to form a multi-tier hierarchical clustering network [**30**]. A number of cluster’s properties such as cluster size and the degree of overlap, which are useful for the management and scalability of the hierarchy, are also considered while grouping the nodes. In the Hierarchical control clustering scheme. Node in the WSN can initiate the cluster formation process. Initiator with least node ID will take precedence, if multiple nodes started cluster formation process at the same time. The algorithm proceeds in two phases: Tree discovery and Cluster formation. The tree discovery phase is basically a distributed formation of a Breadth-First-Search (BFS) tree rooted at the initiator node. Every node updates its sub-tree size when its children sub-tree size change. The cluster formation phase starts when a sub-tree on a node crosses the size parameter.

LEACH - Low Energy Adaptive Clustering Hierarchy

The LEACH protocol is an elegant solution to the data aggregation problem where clusters are formed in a self-organized manner to fuse data before transmitting to the base station or sink [31]. In LEACH, a designated node in each cluster, called the cluster head is responsible for collecting and aggregating the data from sensors in its cluster and eventually transmitting the result to the base station. An improved version of LEACH, called LEACH-C [32] does cluster formation at the beginning of each round using a centralized algorithm by the base station.

HEED - Hybrid Energy-Efficient Distributed Clustering

HEED **[33]** is a distributed clustering scheme in which cluster head nodes are picked from the deployed sensors. HEED considers a hybrid of energy and communication cost when selecting cluster head nodes. Unlike LEACH, it does not select cell-head nodes randomly. Only sensors that have a high residual energy can become cell-head nodes. In HEED, each node is mapped to exactly one cluster and can directly communicate with its cluster head.

CHAPTER 4 - Methodology

4.1. Approaches for role mapping

We consider two major approaches for role mapping.

I. **When the network topology is known**

In this scenario we assume that the network topology is available before hand , the network has a regular structure and nodes are location aware and nodes are associated with coordinates. We assume that the network nodes are stationary and they have an idea of what the location of their neighbors is.

II. **When the network topology is not known**

In this scenario we assume that the network topology is not available before hand and the network has a random asymmetric structure and nodes are not location aware. We assume that the network nodes are stationary and they have no idea of what the location of their neighbors is.

We propose multiple methods for the above stated approaches.

When the network topology is known we can use the following techniques for role mapping

I. No fusion without Backbone.

II. No fusion with Backbone.

III. Incremental mapping of Fusion Nodes in a grid.

When the network topology is not known we propose the following techniques for role mapping

IV. Clustering techniques.

V. Incremental mapping of Fusion Nodes with clusters.

4.2. No fusion without Backbone

In no fusion without Backbone all the sensing nodes send individual data streams to the sink node independently and no data fusion takes place within the sensor network. No Fusion without Backbone is a case of many-to one communication. The objective of implementing a basic sense and send protocol is to set a baseline for other role mapping schemes. Routing without fusion would involve more data traffic in the network as every source node in the wireless sensor network would aim to send data to the sink through the shortest possible path available between the source node and the sink node. No fusion or aggregation of any type would be implemented within the network. The task graph is completely mapped to the sink node. The sink node would be responsible for fusing and aggregating all the data that it gathers from its respective source nodes. Thus, in this approach of sending data to base station without any fusion occurring in the network, one can expect the life time of the network to be considerably reduced when compared to other methodologies which take advantage of local heuristics to perform fusion and aggregation of data streams internally in the network.

This approach works by constructing a minimum weight tree rooted at the sink. Flooding is one of the most widely used data dissemination techniques used for communication in sensor networks but these methods have some inherent limitations. In flooding related techniques each sensor node broadcasts data packet to its neighbors and this process continues until the data packet reaches the destination node. However, the problem with flooding is that it results in unrestricted creation of duplicate packets throughout the network, thus leading to packet congestion and energy consumption.

The protocol for routing data messages from any sensor to the base station in a sensor network is as follows. The basic sense and send protocol for sending source messages to the sink maintains an incoming minimum weight tree rooted at the base station. The tree is constructed as follows.

I. Every sensor in the wireless sensor network topology is assigned a unique identifier or local address that distinguishes a sensor from other nodes in the network.

II. Each and every sensor node calculates the address identifier of its neighboring nodes based on its own identifier.

III. The sink node periodically broadcasts a beacon message to all its neighbors. All the neighbors within the broadcast range can receive a beacon message.

IV. The source identifier of the beacon message as well as the level or hop count from the Sink node to the node sending the beacon message is embedded within the message.

V. Whenever a node receives a beacon message with a level less than the current level of the sensor node, it updates its parent in the tree as the source of the beacon message, increment the hop count and broadcast the beacon message. Thus, a logical minimum weight tree is formed over the sensor network, where every node chooses the node that provides shortest path to the sink node as its parent in the logical tree.

VI. The Source nodes start sending data through directed uni-cast messages to its parents in the logical tree formed.

VII. Whenever a node receives a uni-cast data message it forwards the data message to its parent node in the tree.

```
TreeFormation()
{if Node receives packet for the first time then
    Mark Node as received
    Parent = Sourceofpacket
    Source = Node
    Increment Level Field
    Rebroadcast packet
end if
else if Node receives packet
    if Received Level < Current level
        Parent = Sourceofpacket
        Source = Node
        Increment Level Field
        Rebroadcast packet
    end if
end if
 }
```

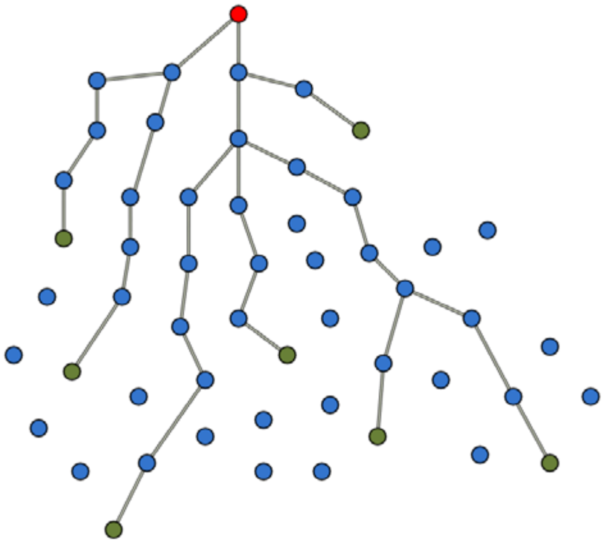


Figure 4.1 Generic routing without Backbone

The above diagram [Fig. 4.1] illustrates routing source data to the sink node without any fusion taking place within the network. Each and every source node tries to route data along the shortest path to the source node.

4.3. Backbone or Grid

Backbone or Grid in a sensor network topology can be considered as a set of high energy nodes that are placed uniformly over the sensor field. The intuition behind grid based routing is to prolong the network lifetime by routing data over a set of high performance nodes. Fully charged battery powered sensor nodes are randomly placed in the field with a sink and a set of sensor nodes. The sensor field is divided into square-shaped grids of user defined grid size. Fig 4.2 illustrates the arrangement of the nodes in the form of a grid.

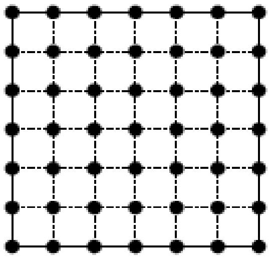


Figure 4.2 Grid arrangement

The objectives of using a grid are to

I. Extend network lifetime by only routing through grid nodes.

II. Maintain network connectivity and prolong network partition time.

Note: The terms Grid Nodes and Backbone Nodes would be used interchangeably to denote the high energy nodes present in the network.

4.4. No fusion with Backbone

In no fusion with Backbone all the sensing nodes send individual data streams to the sink node independently over a network of Backbone nodes also termed as grid nodes and no data fusion takes place within the sensor network. No Fusion with Backbone is a case of many-to one communication. As detailed earlier routing within a network by flooding would result in more data traffic as every source node in the wireless sensor network would aim to send data to the sink through the shortest possible path available between the source node and the sink node. In this approach no data fusion would be implemented within the network. The Task graph is completely mapped to the sink node. The sink node would be responsible for computation and aggregation of data that it gathers from its respective source nodes.

The protocol for routing data messages from any sensor to the base station in a sensor network over a set of Backbone nodes is as follows. The basic sense and send protocol for sending source messages to the sink maintains an incoming minimum weight tree rooted at the base station. The tree construction over the Backbone is detailed below.

Every sensor in the wireless sensor network topology is assigned a unique identifier or local address that distinguishes a sensor from other nodes in the network. The sensor nodes in the network form a logical two dimensional grid in the network topology.

I. Each and every sensor node calculates the address identifier of its neighboring nodes based on its own identifier.

II. The sink node periodically broadcasts a beacon message to all its grid neighbors. All the neighbors within the broadcast range can receive a beacon message but only the Backbone nodes can act on the beacon message.

III. The source identifier of the beacon message as well as the level or hop count from the Sink node to the Grid node sending the beacon message is embedded within the message.

IV. Whenever a Backbone node receives a beacon message with a level less than the current level of the sensor node, it updates its parent in the tree as the source of the beacon message, increment the hop count and broadcast the beacon message. Thus a logical minimum weight tree is formed over the Backbone nodes of the sensor network where every Grid node chooses the Grid node that provides shortest path to the Sink node as its parent in the logical tree.

V. The non-grid nodes within the sensor field choose the nearest Backbone nodes. In case there are multiple Backbone nodes within the vicinity of a a non-grid node it chooses the Grid node closest to the Sink.

VI. The Source nodes start sending data through directed uni-cast messages to its Backbone parents in the logical tree formed.

VII. Whenever a Backbone node receives a uni-cast data message it forwards the data message to its parent node in the tree.

```
TreeFormationGrid()
{if Node receives packet for the first time then
Mark Node as received
Parent = Sourceofpacket
Source = Node
Increment Level Field
If Node is GridNode
Rebroadcast packet
end if
end if
else if Node receives packet
if Received Level < Current level
Parent = Sourceofpacket
Source = Node
Increment Level Field
If Node is GridNode
Rebroadcast packet
end if
end if
end if
}
```

4.5. Incremental mapping of fusion nodes

The role mapping problem

Given N = (Vn, Em), namely, the network topology, and T = ($V't, E't$), namely, the task graph, and an application-specific cost metric M, the goal is to find a mapping f $: V't \rightarrow Vn$ that minimizes the overall cost C. Here, Vn represents nodes of the sensor network and Em represents communication links between them. In the task graph, $V't$ represents fusion functions (filter, data fusion, etc.) and $E't$ represents flow of data between the fusion points. A mapping f : $V't \rightarrow Vn$ generates an overlay network of fusion points to network nodes; implicitly, this generates a mapping l: $E't \rightarrow \{e|e \in En\}$ of data flow to communication links. The focus of the role assignment algorithm is to determine f. Here we assume that the global network topology is not known and must be discovered.

Phases in mapping a task graph to a topology

a) Role mapping and incremental reinforcement for optimization, phase

In the role mapping phase an approximate assignment of the fusion points is made to the Sensor network grid nodes. If there is data contraction taking place then the fusion node placement is made in such a way that the mapped grid node is closer to the source nodes and away from the Sink node but in a direction towards the sink node. Initial placement can be obtained by calculating a weighted centroid of the Source and the Sink nodes. The weights are user defined and can be calculated as a function of the size of the network and the distance between the nodes. Initial placement of the fusion node plays a role in finding the optimum placement of the fusion node. The placement of the fusion node is refined iteratively so as to attain optimal role assignment of the fusion node onto the network nodes. Initially the mapping of the Source and Sink nodes is known and this is the only role mapping that would remain constant during role assignment. The optimal placement of the fusion point is not only dependent on the location of the child nodes but also dependent on the mapping of the parent nodes onto the sensor network topology. Thus, the initial role mapping is performed in a bottom-up manner from the child node to the parent node and the incremental reinforcement is performed recursively in top-down manner from the parent node to the child node. Thus, the initial role mapping and incremental refinement phases are performed in an interleaved manner so as to achieve optimal placement of fusion points onto the network. In the incremental reinforcement phase a grid node that has been assigned a fusion role informs its neighbors in a user defined region of the data fusion cost. Upon receiving this message all the neighbors compute their own data fusion cost and decide if they are more suitable to play the fusion role than the current fusion node. If a neighboring node decides that it is more suitable for performing the fusion role, it intimates the current fusion node of the data fusion cost. A role transfer is performed by the Fusion node to the neighbor node with the least cost of data fusion. After every role transfer the state of the network moves towards the optimum role mapping state. The network reaches a constant state after the role mapping phase is performed. The role mapping phase can be terminated explicitly after a user defined number of iterations are completed.

b) Maintenance phase

In the maintenance phase the residual energy levels of the nodes are constantly monitored and a role transfer operation is performed after a threshold is breached. The incremental reinforcement can be performed after a certain number of role transfers have been performed.

Algorithm mapping fusion nodes to the nodes

```
If Node is event source & data ready
forward data to Geographical data Aggregator

If Node is data Aggregator
Aggregate data and forward data

If Node receives data From Multiple Sources && data items map task graph && node not already a fusion node && other fusion nodes present

calculate data fusion cost

{If data fusion cost <threshold
Node.Fusion_region=true
Fuse data and process data till we reach next fusion point in task graph
```

```
forward this data into network
}

else forward data

If Node receives fused data && is a fusion node for that fused data
Calculate data transfer cost from previous source of fused data
Send data transfer + data fusion cost back to the previous fusion node

If fusion node receives feedback from forward fusion region calculate average cost of fusion and data transfer
Start Timer
Publish this cost in fusion region
Timer expires Set Node as Fusion Node'

If node receives average Fusioncost && Received Cost < Average Fusion cost
Send Negative Reply

Mapping the intermediate computation stages Onto the intermediate Nodes

Node is in Fusion Channel
while(incremental Computation overhead <data transfer cost+ computation cost at next node)
Assign task to Node
else
propogate Task to next node in fusion channel
If Node.dataReady==true
{
If Node.isSource==true  And Node.isAggregator==false
Node.Send(data,Aggregator_id)
else
If Node.isAggregator==true
Node.Aggregate(Data_Set)
}

If Node.isAggregator==true And Node.dataReceived==true
Node.Aggregate(Data_Set)
Node.Send(Data,Metric)

If Node.dataReceived==true And Received_data_Metric > Node.Threshold
{
```

```
Node.setDataType=True
Node.Send(data,Metric)
else
Node.dropData
}


If Node.dataTaskMap==True
{
If(Node.fusionCost(Data)<threshold)
Node.fusionRegion=true
Node.Send(fusedData,Metric)
Node.sendFeedBack(Cost)
}

If Node.receivesFeedBack==True
Node.publish(ForwardFusionNode)

If Node.fusionNode==false
Node.SendFusionNodeId(FusionNode,Metric)

If  Node.fusionRegion==true And Node.ReceiveFusionNodeId==true
Node.setFusionNodeId(FusionNode)
else
If  Node.fusionRegion==false And Node.ReceiveFusionNodeId==true
Node.SendFusionNodeId(FusionNode,Metric)
```

4.6. Heuristics for routing on Backbone

In this section we describe the heuristics for routing on a Backbone and come up with algorithms for efficient routing in the presence of a grid structured Backbone.

Assumption

For routing on a grid we assume that two data nodes and one fusion nodes are identified .The data nodes know the position of each other and the fusion node. Let the coordinates of the Data Nodes be (x1,y1) and (x2,y2) and the Fusion node Coordinates be (xf,yf)

Case1

All 3 lie on the same row or column:

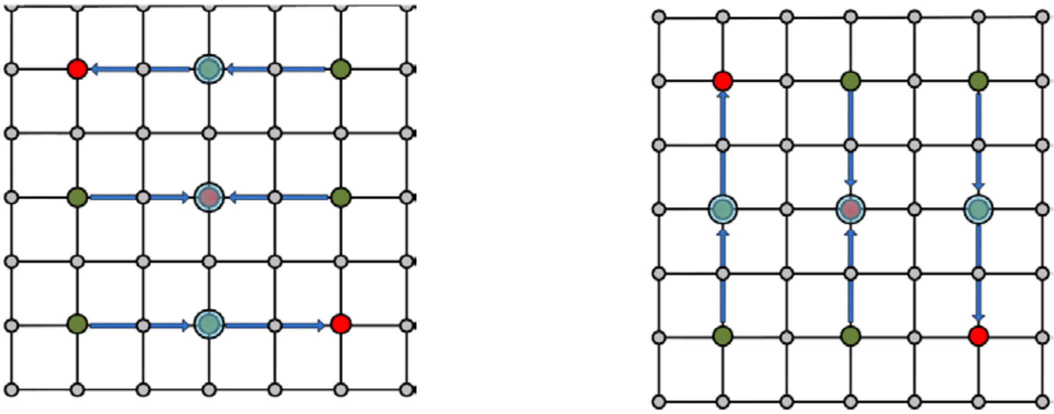


Figure 4.3 Grid routing when all three are in same row or column

When all the three nodes lie on the same row or column then we send data directly to the sink node [Fig. 4.3].

```
If (x1==x2==xf)
{
y1<yf<y2 fusion node =(xf,yf)
else
if |y1-yf| <|y2-yf|
fusion node = (x1,y1)
else
fusion node =(x2,y2)
}
```

```
Else
If (y1==y2==yf)
{
x1<xf<x2 fusion node =(xf,yf)
else
if  |x1-xf| <|x2-xf|
fusion node = (x1,y1)
else
fusion node =(x2,y2)
}
```

Case 2

All 3 lie in a straight line and the fusion node is in-between the two data nodes

if $|x1-xf| + |x2-xf| + |y1-yf| + |y2-yf| == |x1-x2| + |y2-yf|$

reach fusion node

Case 3

The fusion node lies on the same side of the data nodes.

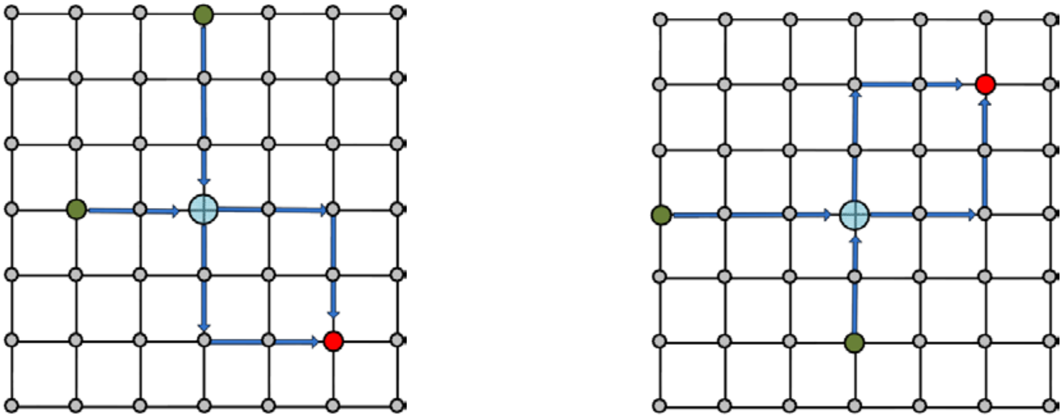


Figure 4.4 Grid routing when Source nodes lie on the same side of Fusion node

There could be two scenarios

a) Data nodes are on either side of the fusion node [Fig. 4.5].

b) Data nodes are on the same side of the fusion nodes [Fig. 4.4].

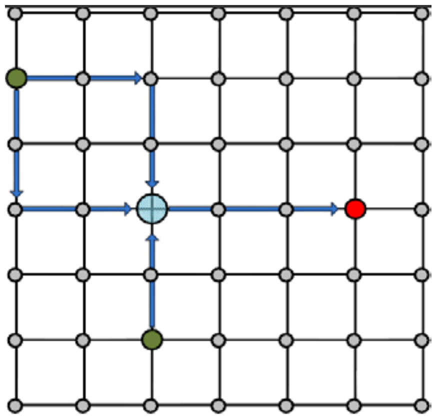


Figure 4.5 Grid routing when Source nodes lie on the either side of Fusion node

```
if  $|x1-xf| +|x2-xf| + |y1-yf| +|y2-yf| > |x1-x2| +|y2-yf|$
 {
        If  $x1<xf<x2$ or $y1<yf<y2$
           {//data points are on either side of the fusion point
                   If $|x1-x2| > |y1-y2|$
                        { Travel along x-axis
        if $|y1-yf| < |y2-yf|$
                            New fusion point =$(xf,y1)$
                          Else
                         New fusion point =$(xf,y1)$
                        }
                 else
                      { Travel along Y-axis
        if $|x1-xf| < |x2-xf|$
                            New fusion point =$(x1,yf)$
                          Else
```

```
                       New fusion point =(x2,yf)
                      }
             }
        Else
            {//Data points are on the same side of the fusion point
                If(|x1-x2| >|y1-y2|)
                {  if(|y1-yf| >|y2-yf|)
new fusion point = (x1,y2)
                else
new fusion point = (x2,y1)
}
Else
{  if(|x1-xf| >|x2-xf|)
new fusion point = (x2,y1)
                else
new fusion point = (x1,y2)}
}
```

4.7. Clustering

In this section we specify the need for clustering in random topology and describe the distributed implementation of density based clustering methods.

Clustering the nodes in a sensor network can be an effective technique for optimizing network lifetime, achieving efficient scalability, and load balancing. In this section we propose energy-efficient approaches for clustering nodes in ad-hoc sensor networks. Clustering can be helpful when the topology of a network is not known and when an application has to scale over a large number of nodes. Some Clustering techniques are inherently distributed and can be adapted in a sensor network environment. Applications that need efficient data aggregation and fusion in a scalable environment are natural candidates for using clustering approaches for example the average temperature of a topological region can be calculated efficiently by calculating the temperature of the clusters associated with the topological region, this approach not only eliminates redundant data transmission but also ensures that some amount of computation and data processing shifts within the network thus making the application more responsive. Clustering can not only be utilised for data aggregation and data fusion but also for routing in the network.

The following images[Fig. 4.6] illustrate the need for clustering in a single hop environment where messages are sent directly to the sink node from the individual sensing nodes. By clustering the sensor field and aggregating or fusing data in the individual clusters a lot of messages can be saved this approach not only improves the residual energy levels in the network thus extending the network life time it also makes the approach more scalable to larger networks.

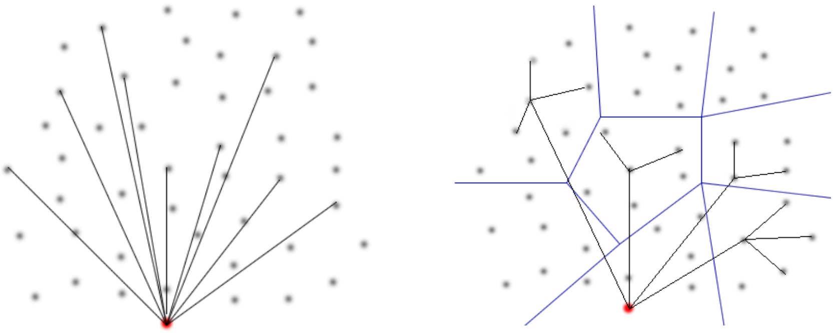


Figure 4.6 Single hop without clustering and Single hop with clustering

Similarly the following images[Fig 4.8] illustrate the advantage of clustering in a multi-hop environment where messages are routed to the sink node from the individual sensing nodes. By clustering the sensor field and aggregating or fusing data of the individual clusters a lot of messages can be saved. Clustering can simplify routing in a multi-hop environment by dividing the routing function into intra-cluster routing and inter-cluster routing. This design can help aggregate data within the clusters and also fuse related data among clusters. Clustering can be an efficient technique for in-network data processing to extend the overall lifetime of a sensor network.

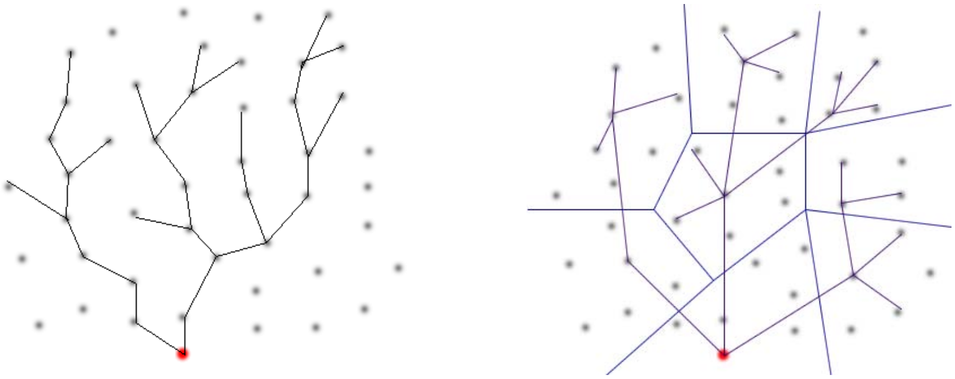


Figure 4.7 Multi-hop without clustering and Multi-hop with clustering

The following are the advantages of Clustering as proposed in the above approaches

I. Clustering reduces resource contention thus increasing network life time

II. Cluster states give a partial picture of the Network state and aggregating Cluster states the overall network can be captured.

III. Clustering provides an efficient way to hierarchically structure a network topology and provides a clean and elegant way to logically represent a network whose topology is not known.

IV. Efficient network routing can be achieved by reducing the routing problem as routing through an overlay of clusters which effectively has a relatively smaller network diameter.

Requirements for a good clustering technique for a sensor network application

I. Clustering must be distributed; it should not be either initiated or terminated by a central node.

II. It should not make any assumption about the topology, density or distribution of nodes over the Sensor network.

III. All the nodes are assumed to be of similar capabilities.

IV. The clustering mechanism must terminate within fixed number of iterations and must be independent of the topology, size or the scale of the network.

V. Clustering should incur limited and negligible overhead and maintenance of clusters should not impede with the regular working of the sensor network application.

VI. The result of clustering should be a uniformly logically segmented network.

4.8. Formal definition of the network clustering problem

Let the Network topology consist of K nodes with identical capabilities. The objective of the clustering problem is to segment the topology logically into a set of "K" clusters; each cluster is associated with a cluster head and a set of core nodes. After clustering each node Ni in the network is mapped at most to one of the K clusters. The K clusters have disjoint membership and must cover the entire network topology. Within each cluster every node is either a core node or a dependent node which is in direct communication range with one of the core nodes. One of the core nodes is selected as a Cluster head. Thus every node in a connected network is either a core node or a border node. A node which is not in direct communication range with the core nodes is classified as an outlier.

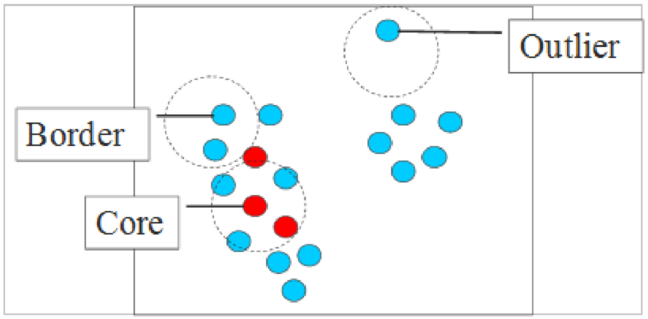


Figure 4.8 DBSCAN Node varieties

For a clustering approach to be efficient the following requirements must be met

I. Clustering must be completely distributed and every node takes its decisions independently and solely based on local information.

II. Regardless of the scale and diameter of the network clustering must terminate.

III. After each iteration every node is either a core node, border node or an outlier node.

IV. Clusters must be evenly distributed over the network.

In this section, we describe our two clustering protocols in detail. First, we define the parameters used in the clustering process. Second, we present the protocol design and pseudo-code. Finally, we prove that the protocol meets its requirements.

4.9. Density-Based Spatial Clustering of Applications with Noise (DBSCAN):

DBSCAN is a density based clustering algorithm which can detect clusters of arbitrary shape and size in diverse settings [34]. Density of a cluster can be measured in various ways ranging from simple metrics like the count of data object in the cluster to more complex functions on the count and location of data items in the cluster region. Basic idea behind DBSCAN clustering approach is that clusters are defined as dense regions separated by sparsely populated regions. DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. In DBSCAN a data object can either be classified as a cluster member or as noise. DBSCAN is based on the concepts of “density reachability” [Fig. 4.11] and “density connectivity” [Fig. 4.12] both measured in terms of local distribution of the nearest neighbors.

Conceptually, the Nodes in a network can be classified into three classes:

I. **Core points**. The core points of a cluster lie interior to the cluster. For a data point to be a core point it has to have a minimum number of points in its E-neighborhood i.e. if the number of data points exceeds the core object threshold value within a given radius or E-neighborhood around the point then such a data point is considered to be the Core object. If two core points are present in each other’s neighborhoods, then the core points belong to the same cluster and are considered to be directly density reachable.

II. **Density reachable points or a Border point**. A density reachable point is a point in a cluster that is not a core point, i.e. it has at least one core object in its neighborhood but there are not enough data points in its neighborhood that exceed the threshold value to be a core object. Thus, for DBSCAN, a cluster is the set of all core points whose neighborhoods transitively connect them together, along with some border points.

III. **Noise points**. A noise point is any point that is not a core point or a border point.

Any border point that is close enough to a core point is put in the same cluster as the core point .Noise points are usually discarded.

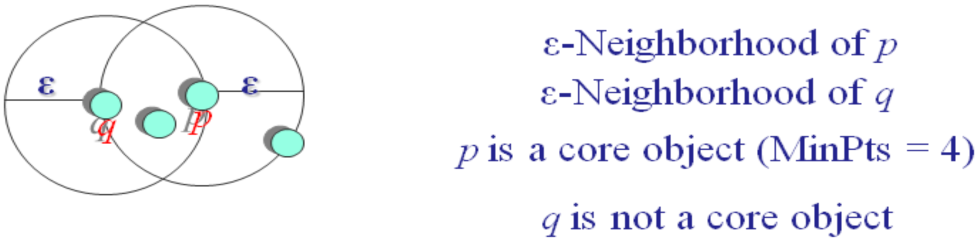
E-Neighborhood

Objects within a radius of ε from an object constitute the epsilon-neighborhood of an object [Fig. 4.10].

Core objects

An object is a core object if ε-Neighborhood of the object contains at least MinPts of objects.

The following example [Fig. 4.11] illustrates how core nodes are identified using E-Neighborhood.



ε-Neighborhood of *p*

ε-Neighborhood of *q*

p is a core object (MinPts = 4)

q is not a core object

Figure 4.9 Epsilon neighborhood

Directly density-reachable

An object q is directly density-reachable from object p if q is within the ε-Neighborhood of p and p is a core object. In the above example q is directly density reachable from p as p is a core object .But p is not directly density reachable from q even though it is in ε-Neighborhood of q as q is not a core object.

Density-reachable

An object *p* is density-reachable from *q* with respect to ε and *MinPts* if there is a chain of objects *p1,…,pn*, with *p1=q*, *pn=p* such that *pi+1*is directly density-reachable from *pi* w.r.t ε and *MinPts* for all 1 <= *i* <= *n*. The following example illustrates the concept of density reachability.

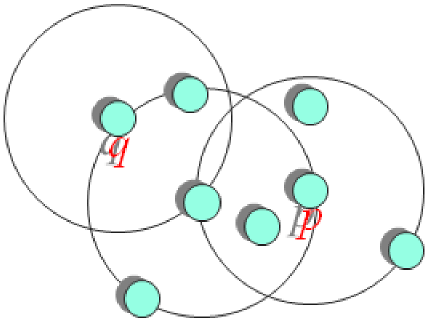


Figure 4.10 Density Reachable

In the above example *q is density-reachable from p* but not vice-versa, *i.e.* Transitive closure of direct density-reachability is asymmetric [Fig. 4.12].

Density-connectivity

Object p is density-connected to object q w.r.t ε and MinPts if there is an object r such that both p and q are density-reachable from r w.r.t ε and MinPts. The following example illustrates the concept of Density-connectivity.

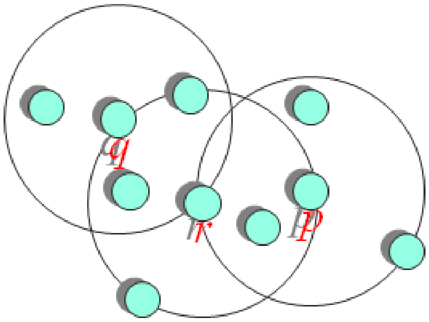


Figure 4.11 Density connectivity

In the above example Figure 4.13 P and q are density-connected to each other by r, i.e. Density-connectivity is symmetric.

Cluster

A cluster **C** in a set of objects **D** w.r.t ε and MinPts is a non empty subset of **D** satisfying the following conditions of maximality and connectivity.

- **Maximality**: For all p, q if p ∊ **C** and if q is density-reachable from p w.r.t ε and MinPts, then also q ∊ **C.**

- **Connectivity**: for all p, q ∈ **C**, p is density-connected to q w.r.t ε and MinPts in **D.**
- **Noise :** Objects which are not directly density-reachable from at least one core object.

Algorithm for DBSCAN:

```
DBSCAN(D, eps, MinPts)
  C = 0
  for each unvisited point P in dataset D
    mark P as visited
    N = getNeighbors (P, eps)
    if (sizeof(N) < MinPts)
      mark P as NOISE
    else
      C = next cluster
      expandCluster(P, N, C, eps, MinPts)
expandCluster(P, N, C, eps, MinPts)
  add P to cluster C
  for each point P' in N
    if P' is not visited
      mark P' as visited
      N' = getNeighbors(P', eps)
      if N' >= MinPts
        N = N joined with N'
    if P' is not yet member of any cluster
     add P' to cluster C
```

Complexity

DBSCAN visits each point of the dataset, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the time complexity is mostly governed by the number of getNeighbors queries. DBSCAN executes exactly one of such queries for each point, and if a sufficiently performant indexing structure is used that executes such a neighborhood query in O(logn), a overall runtime complexity of $O(n \cdot \log n)$ is obtained.

4.10. Structured Clustering Algorithm for Networks (SCAN)

SCAN is a structural similarity based clustering algorithm which can detect clusters of arbitrary shape and size in diverse settings [35]. SCAN performs clustering by trying to identify the structural similarity of nodes. In SCAN nodes with the same structural similarity will be part of the same cluster. In SCAN a data object can either be classified as a cluster member, as noise or Outliers, as Hubs. SCAN is based on the concepts of structural similarity that states that members of same clique have many similar adjacent members irrespective of the size of the clique or cluster. Conceptually, the Nodes in a network can be classified into the following classes

- **Core points**. The core points of a cluster lie interior to the cluster. For a data point to be a core point it has to have a minimum number of points in its E-neighborhood i.e. if the number of data points exceeds the core object threshold value within a given radius or E-neighborhood around the point then such a data point is considered to be the Core object. If two core points are present in each other's neighborhoods, then the core points belong to the same cluster and are considered to be directly density reachable.

- **Hub:** Hubs are nodes that have multiple Core nodes in their neighborhood and these core nodes belong to different structures. Hubs act as bridges to multiple clusters.

- **Direct structurally reachable points or a Border point**. A density reachable point is a point in a cluster that is not a core point, i.e. it has at least one core object in its neighborhood but there are not enough data points in its neighborhood that exceed the threshold value to be a core object. Thus, for SCAN, a cluster is the set of all core points whose neighborhoods transitively connect them together, along with some border points.

- **Outliers or Noise points**. An outlier or noise point is any point that does not belong to any cluster and is not a hub.

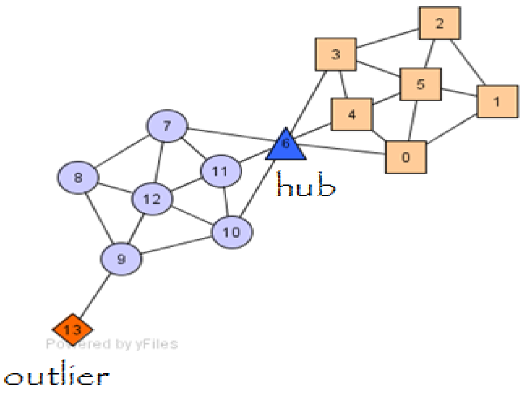


Figure 4.12 Elements of SCAN

Neighborhood

The Neighborhood T(v) of a node is defined as the number of neighbors or number of nodes that are in its communication range .

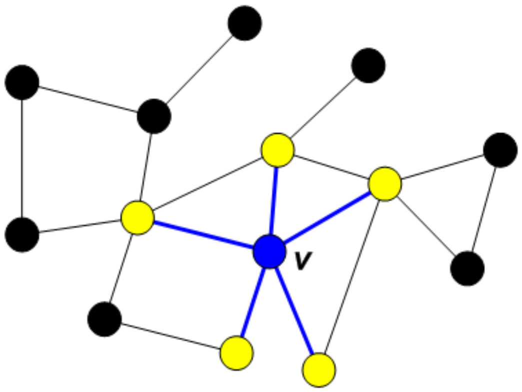


Figure 4.13 Node Neighborhood

In the above image [Fig. 4.15] the yellow nodes represent the neighborhood of node V.

Structural similarity

Structural similarity is a measure of commonality of two adjacent nodes. Structural similarity of two adjacent nodes V, W can be given by

$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

Structural similarity is large for members of same cluster or clique and small for hubs and outliers.

E-Neighborhood

It is the number of adjacent nodes of a Node with a structural similarity above the ε- threshold (read as epsilon-neighborhood).

$$N_{\varepsilon}(v) = \{w \in \Gamma(v) \mid \sigma(v, w) \geq \varepsilon\}$$

Core objects

An object is considered to be a Core object if the ε-Neighborhood of an object contains at least **MinPts** of objects.

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow | N_{\varepsilon}(v) | \geq \mu$$

Directly structure-reachable

An object q is directly structure-reachable from object p if q is within the ε- Neighborhood of p and p is a core object.

$$DirRECH_{\varepsilon,\mu}(v, w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_{\varepsilon}(v)$$

Structure-reachable

An object p is structure-reachable from q with respect to ε and MinPts if there is a chain of objects p1,…,pn, with p1=q, pn=p such that pi+1is directly structure-reachable from pi w.r.t ε and MinPts for all 1 <= i <= n.

Structure-connectivity

Object p is Structure-connected to object q w.r.t ε and MinPts if there is an object r such that both p and q are Structure-reachable from r w.r.t ε and MinPts.

$$CONNECT_{\varepsilon,\mu}(v, w) \Leftrightarrow \exists u \in V : RECH_{\varepsilon,\mu}(u, v) \wedge RECH_{\varepsilon,\mu}(u, w)$$

A cluster C in a set of objects D w.r.t ε and MinPts is a nonempty subset of D satisfying the following conditions of maximality and connectivity.

Maximality

For all p, q if p∈C and if q is density-reachable from p w.r.t ε and MinPts, then also q ∈C.

$$\forall v, w \in V : v \in C \wedge REACH_{\varepsilon,\mu}(v, w) \Rightarrow w \in C$$

Connectivity- For all p, q∈C, p is density-connected to q w.r.t ε and MinPts in D.

$$\forall v, w \in C : CONNECT_{\varepsilon,\mu}(v, w)$$

Noise objects are objects which are not directly structure-reachable from at least one core object.

4.11. Distributed algorithm for DBSCAN and SCAN

```
//Initialise the nodes in sensor network

command result_t StdControl.init()
{
    //each node is initialized randomly to replicate real world
operation
    return call Random.init();
}

//Broadcast a beacon packet to neighbors on starting up

command result_t StdControl.start()
{
    //initializing number of adjacent nodes and the nature of
the node
    Min_Nodes=0;
    Core_Node=False;
    //setting the epsilon neighborhood in terms of the radio
transmission power level
    Node.setPower_Level=epsilon_neighborhood;
    beacon_packet.nodeid=TOS_LOCAL_ADDRESS;
    //broadcast an initial beacon message that lets its presence
known to neighboring nodes in its epsilon neighborhood
    call                       SendBeaconMsg.send(TOS_BCAST_ADDR,
sizeof(uint16_t),beacon_packet);
    startTimer1.time();
    startTimer2.time();

}

//Node receives a beacon message

event ReceiveBeaconMsg.receive(TOS_MsgPtr recv_packet)

{
//each receipt of a beacon node signal indicates the presence of
a node in neighborhood
Min_nodes++;
}

//Neighborhood of a node is determined

event Timer1_fired()
{
```

```
//If Node is a Core Object and satisfies energy constraints

If(Min_nodes > Core_Min && Batt_Power > threshold)
{
    Core_Node=True;
    Core_Object_packet.nodeid= TOS_LOCAL_ADDRESS;
    //transmit a message indicating that the node is a neighboring node
    call SendCoreObjMsg.send(TOS_BCAST_ADDR, sizeof(uint16_t),
Core_Object_packet);
return;
}
}

//Node receives a message from a core object

event ReceiveCoreObjMsg.receive(TOS_MsgPtr recv_packet)

{
//Add the neighboring node to its core object list,list contains
items that are directly density reachable and density reachable
from the node
coreObjList.add(node_id);
}


//Allocating a Uniform cluster identification number to all the
core objects in the cluster
event Timer2_fired()
{
If(Core_Node)
{
//assigning clustered as the id of the coreObject with least
node id in the cluster
    If(coreObjList.Min()< TOS_LOCAL_ADDRESS)
    {
    ClusterId_packet.clustid= coreObjList.Min();
    Cluster_Id= coreObjList.Min();
    call SendClusterIdMsg.send(TOS_BCAST_ADDR, sizeof(uint16_t),
ClusterId_packet);
}
return;
}
}

event ReceiveCoreObjMsg.receive(TOS_MsgPtr recv_packet)
{
If(ReceivedPacket.clustid< Cluster_Id)
```

```
{
 ClusterId_packet.clustid= coreObjList.Min();
    Cluster_Id= coreObjList.Min();
    call     SendClusterIdMsg.send(TOS_BCAST_ADDR,     sizeof(uint16_t),
ClusterId_packet);
}
}
```

4.12. Grid formation and role assignment in parallel

Two objectives to be achieved

- **Map task graph onto network**
- **Coming up with a Backbone grid for network**

Above tasks are performed in parallel

Network is divided into cells at each level. At top level we have only one cell. Cells of upper level are divided uniformly to attain cells of lower level. A cell of the nth level corresponds to “k” cells of the (n+1)th level. Each cell has a corresponding cell head. Each cell has a set of parameters.

Cell Parameters could be:

- Number of nodes
- Number of data nodes
- Amount of Energy available in the cell
- Type of distribution of the data/fusion nodes the cell has
- Type of tree/sub tree associated with the cell
- The height of the sub-tree etc.

Framework

Data node initially forwards data to its corresponding cell head at the lowest level.

If the cell contains all the corresponding partner node(s) of this data item then

Cell head determines location of fusion node

else

cell head passes relevant information to corresponding cell head of its parent cell

Cell head determines the size of backbone grid based on above params

Cell head forwards location of fusion node to the data nodes in its cell

Once fusion node is found repeat the above process from the lowest level.

Assumptions

Network is already divided into cells at each level.

Each cell is assigned a cell head and every member of the cell has knowledge about the cell head.

Cell head has total or partial knowledge about the structure of the task graph.

```
If  Node is event source && data ready
{
          If fusion node not identified
        Node.sendData(CellHead,data);

        Else

        Node.sendBBData(fusionNode,data)

}

If  Node is CellHead && dataReceived
{

        If  PartnerNodes.data present in cell

                  {
                  Identify fusion node
                  For all partner Nodes
                    {
                     partnerNode.Send(FusionNode)
                    }
```

```
                }

        Else

        Node.sendData(CellHead,data);
}


If  Node is not CellHead && dataReceived
{
        Node.sendData(CellHead,data);
}


If Node is CellHead && Source nodes identified
{

        Calculate grid size for each child cell
        Identify BB Nodes
}


 If(FusionNode==True And DataReceived ==True)
FuseData(ReceivedData)

If Node is Fusion Node And fused data ready
{

         If next  fusion node not identified
        Node.sendData(CellHead,data);
```

```
    Else

    Node.sendBBData(fusionNode,data)

}
If Node is BackBone Node And Receives data
{

    Node.sendData(Dest);
}
```

CHAPTER 5 - Experimental setup

In this chapter we discuss the implementation environment and experimental set up for the various approaches detailed previously. We implemented the data fusion and in-network aggregation protocols in the nesC language which is an extension of C language designed for implementation in the TinyOS environment. TinyOS is an operating system that is designed to manage the operation of a variety of mote devices as well as the sensors attached to them. TinyOS also provides a networking stack to allow the motes to form an ad-hoc network. The protocol execution is simulated on the TOSSIM discrete event simulator designed to simulate sensor networks that use the TinyOS operating system. For visualisation, TinyViz a Java based GUI front end to TOSSIM is used. Plug-ins for TinyViz which can interact with the TOSSIM simulator are also utilised. Plug-ins for controlling the power model and monitoring the messages transmitted between nodes are used primarily. In our experiments we set the communication range of each sensor to 10 m and assume bi-directional links.

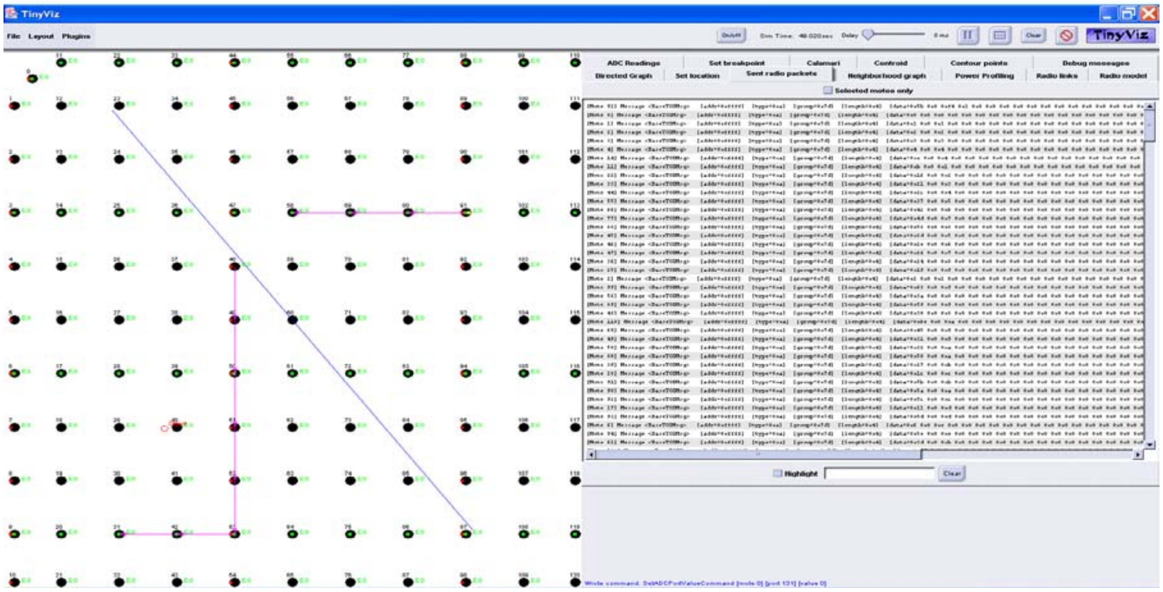


Figure 5.1 TinyViz

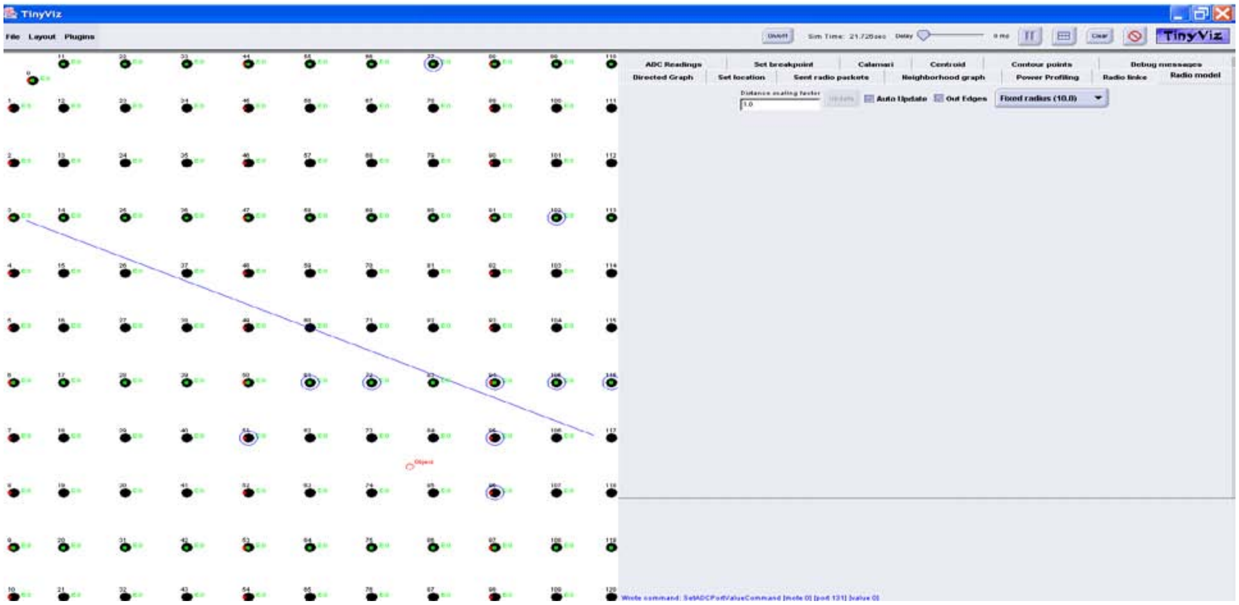
The Figure 5.1 shows a snapshot of the TinyViz visualisation environment. The directed arrows represent messages being passed between the nodes. The messages sent are shown in the right window pane.

We use a uniformly distributed sensor network topology for the set of protocols implemented for a structured topology and use a random walk generated topology for the set of approaches when the topology of the sensor network is not known.

5.1. Set up when topology is known

The TOSSIM simulation environment consists of 121 nodes spread evenly in an area of 100 X 100 sq units. The transmission power of each node was restricted to 10 units [Fig. 5.2]. The performance of the data fusion approaches were measured over same topology but by varying the location of the sink node and the structure of the task graph that has to be mapped onto the wireless sensor network topology. Thus by spacing the nodes uniformly and maintaining a power transmission range of 10 units we ensure that a completely connected network is generated. A Sensor network is considered to be completely connected if in a multi-hop environment every node can reach any other node in the network topology.

In case of approaches where the concept of Backbone is not used all nodes are considered to be the sensor nodes and the following figure shows one such topology [Fig. 5.3].



Distance scaling factor 1.0

☐ Auto Update ☐ Out Edges

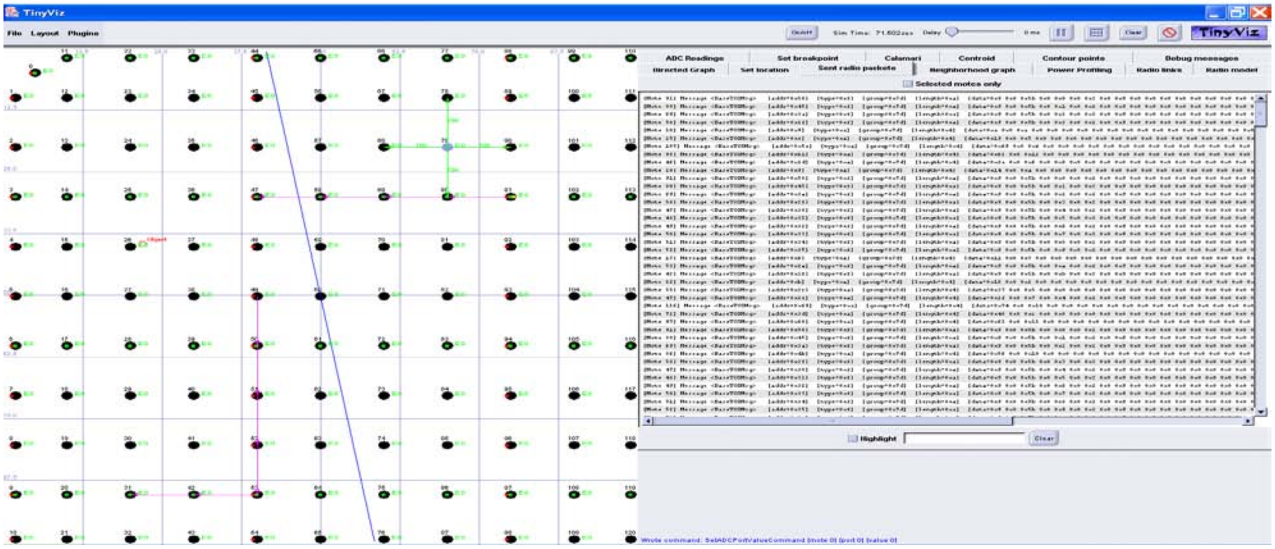
Fixed radius (10.0)

Write command: SetADCPortValueCommand [mote 0] [port 131] [value 0]

Figure 5.2 TinyViz Experimental setup for Generic approach

In the above Figure 5.2 Node 0 is considered to be the sink node and no Backbone is overlayed over the sensor network topology.

In case of approaches where a Backbone is overlayed over the sensor network topology, the performance of the approaches is evaluated by varying the size of the grid. The Backbone nodes are uniformly distributed over the sensor field and are along interleaving rows and columns of the sensor network.



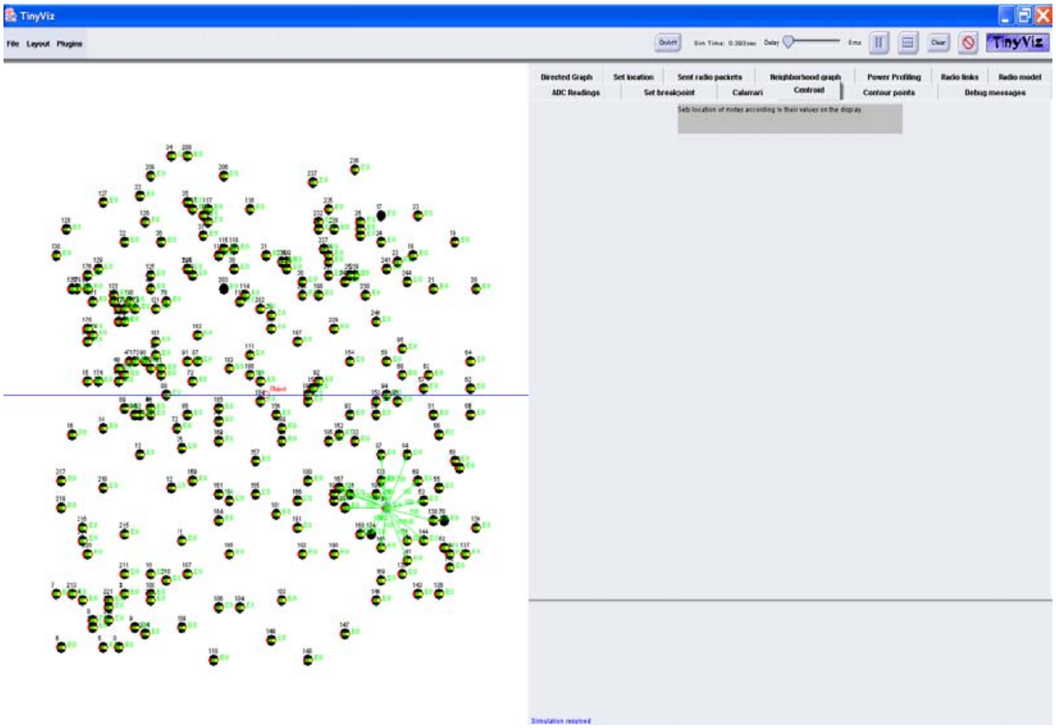
Write command: SetADCPortValueCommand [mote 0] [port 0] [value 0]

Figure 5.3 TinyViz Experimental setup with Backbone Grids

The above figure 5.3 shows a grid topology with interleaved rows and columns used as Backbone for the network. The performance of Backbone oriented approaches are measured for grid sizes of 2X2, 4X4 and 6X6.

5.2. Set up when topology is not know

To measure the performance of clustering approaches where the topology of the network is not known we consider a randomly generated sensor network topology consisting of 245 nodes spread over an area of 100 X 100 square units [Fig. 5.4]. The nodes in the network are provided with a power transmit radius of 10 units and the communication is bi-directional. To generate a completely connected network a random walk algorithm was used to generate a topology in which every node is connected to any other node in the network. Even though the TOSSIM simulator generates a random topology it does not ensure that the wireless sensor network topology generated is connected. To avoid this, a multi seeded random walk algorithm ensures a completely connected network. Thus random walk approach to generate a sensor network topology ensures a large connected network of dense nodes with no global coordinate system. The number of seeds and the initial location of the seeds are user tunable parameters.



Sets location of motes according to their values on the display.

Simulation resumed

Figure 5.4 TinyViz Experimental setup for Random walk generated Topology

Above figure 5.4 shows a sensor network topology where the nodes are placed randomly in the network field.

5.3. Visualisation of clusters

MATLAB is used to visualise the results of distributed clustering approaches [Fig. 5.5]. The following figures show two such results.

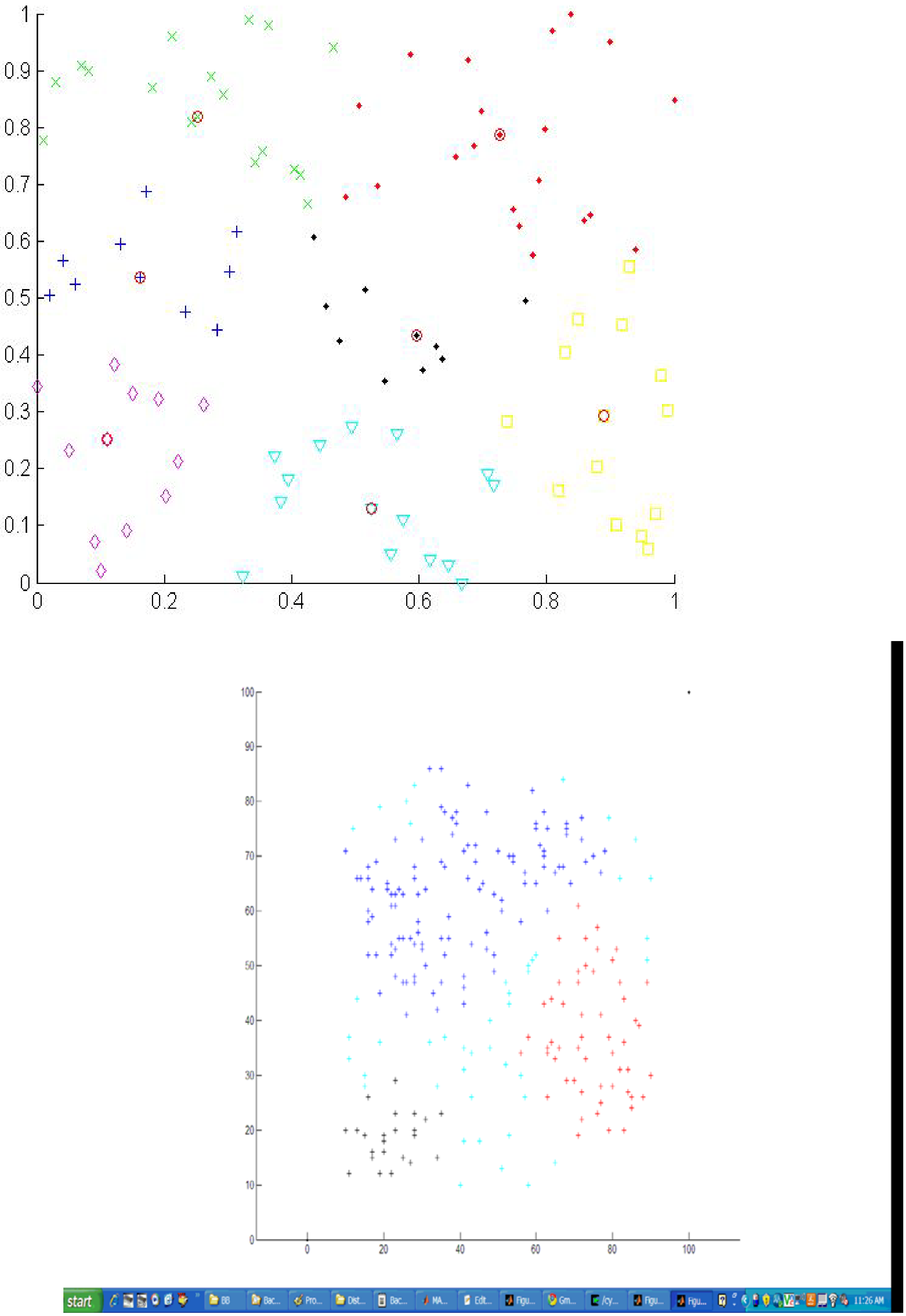


Figure 5.5 Cluster formation

CHAPTER 6 - Results

In this chapter we discuss the testing strategies for various algorithms discussed in the previous chapters. We use the hop-count distance metric as a measure of performance of the various algorithms .The lower the aggregate hop-count the better is the performance of an algorithm for an experimental setup. We present the results for the following experiments we conducted for the various protocols proposed earlier.

- **Effect of structure of task graph on fusion strategy**

  We measure the performance of various strategies for balanced and unbalanced task graphs of varying sizes
- **Effect of Grid size on fusion strategy**

  We measure performance characteristics of various strategies by varying the alignment and the size of the grid.
- **Effect of clustering parameters and clustering mechanisms**

6.1. Test cases for known topology

When Topology is known the approaches are compared for varying sizes of balanced and unbalanced trees. In each test case a source will generate a distinct numbered data packet and data is transmitted according to the fusion strategy. All the task graphs considered are binary trees and source nodes are selected under the assumption that source nodes that are closer in task graph are topologically closer in the network. In case of the generic routing with no Backbone all the data packets from the source nodes are routed in individual routes to the sink node and as there is not in-network data aggregation and fusion this strategy can result in data packets taking different routes to the sink. A worst case scenario for this kind of data transfer scheme would be the case where multiple leaf nodes of the task graph are mapped onto a single source node of the Sensor network. The performance of the strategies measured in terms of hop-count increases as we move away from a generic approach towards Fusion with Backbone.

6.2. Effect of task graph on performance

The intuition behind looking at the performance metrics for a balanced and unbalanced tree is to interpret how choice of a task graph affects the performance and choice of strategy .In general for any given approach the fusion strategy works better for an unbalanced tree when compared to a balanced tree. Figure 6.1 shows the comparison in number of messages transmitted for balanced trees of increasing size for various strategies.

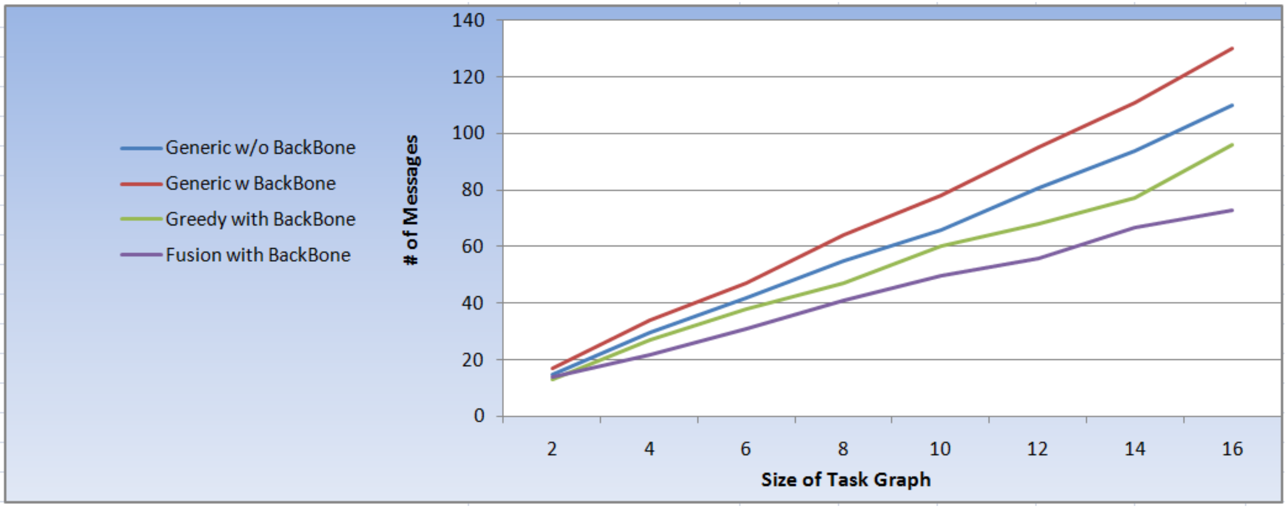


Figure 6.1 Results for Balanced tree

Figure 6.2 shows the comparison in number of messages for unbalanced trees of increasing size for various strategies.

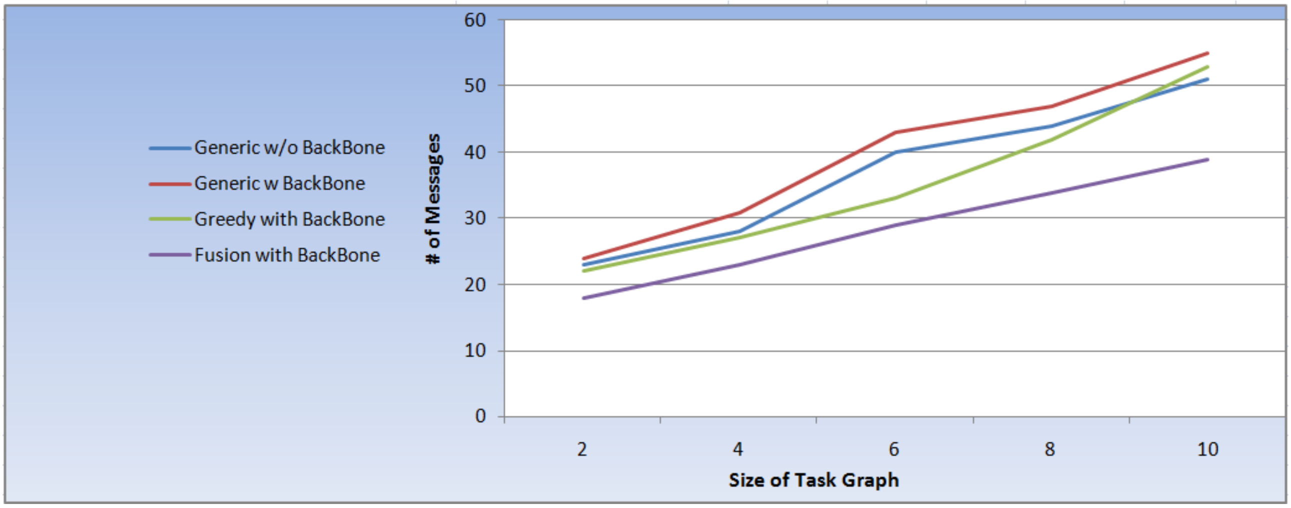


Figure 6.2 Results for Unbalanced tree

It can be seen that incremental fusion strategy performs well irrespective of the nature of the task graph when compared to generic and greedy approaches. Thus in-network data aggregation can help optimise the number of messages transmitted extending the effective lifetime of the network.

6.3. Effect of grid size on performance

It is important to measure the influence of the grid size of the interleaved Backbone network overlayed over the sensor network. For both the Greedy with Backbone and Fusion strategies as the grid size increases the performance of the strategies measured in terms of hop-count decreases. Thus we can conclude that better the density of the Backbone network better is the performance of the in-network aggregation strategy.

Figure 6.3 shows the performance of Greedy fusion approach in number of messages for various grid sizes over various sizes of trees. We can see that as the grid size increases the hop count increases for task graphs of different sizes.

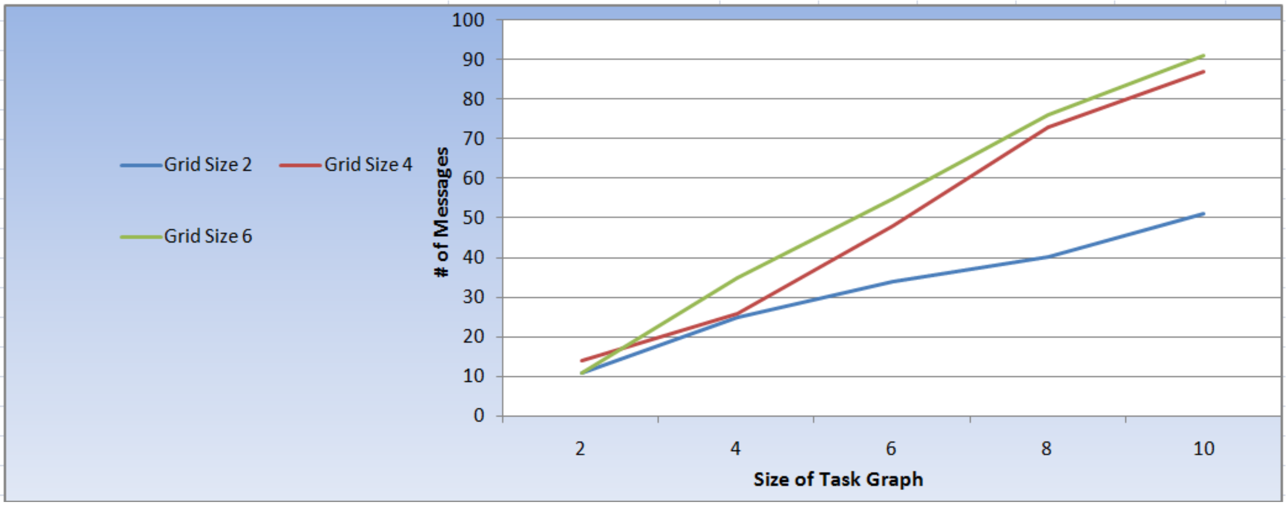


Figure 6.3 Results for Greedy Fusion with varying Grid size

Figure 6.4 shows the performance of Incremental fusion approach in number of messages for various grid sizes over various sizes of trees. We can see that as the grid size increases from 2 to 4 the hop count increases for task graphs of different sizes. But the same is not true for Grid size of 6 as the location of the Source and fusion nodes with respect to the relative position of the Backbone node plays a role.

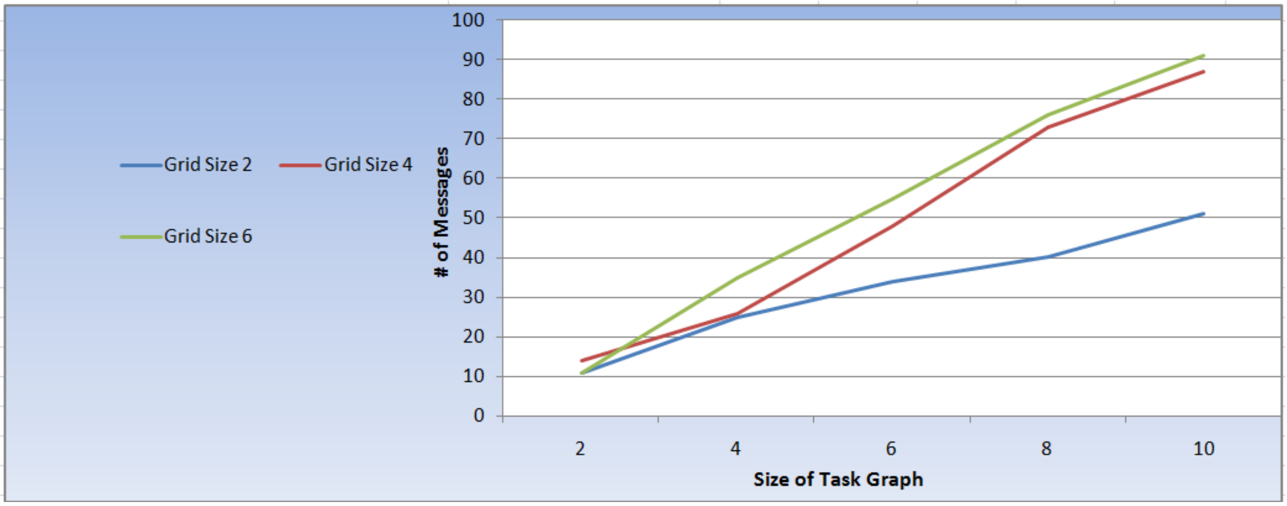


Figure 6.4 Results for Incremental Fusion with varying Grid size

Figure 6.5 compares the performance of Greedy fusion and Incremental approaches in number of messages for various grid sizes over various sizes of trees. We can see that as the grid size increases the hop count increases for task graphs of different sizes and the performance of Incremental fusion strategy is better than Greedy fusion for a given grid size and task graph.

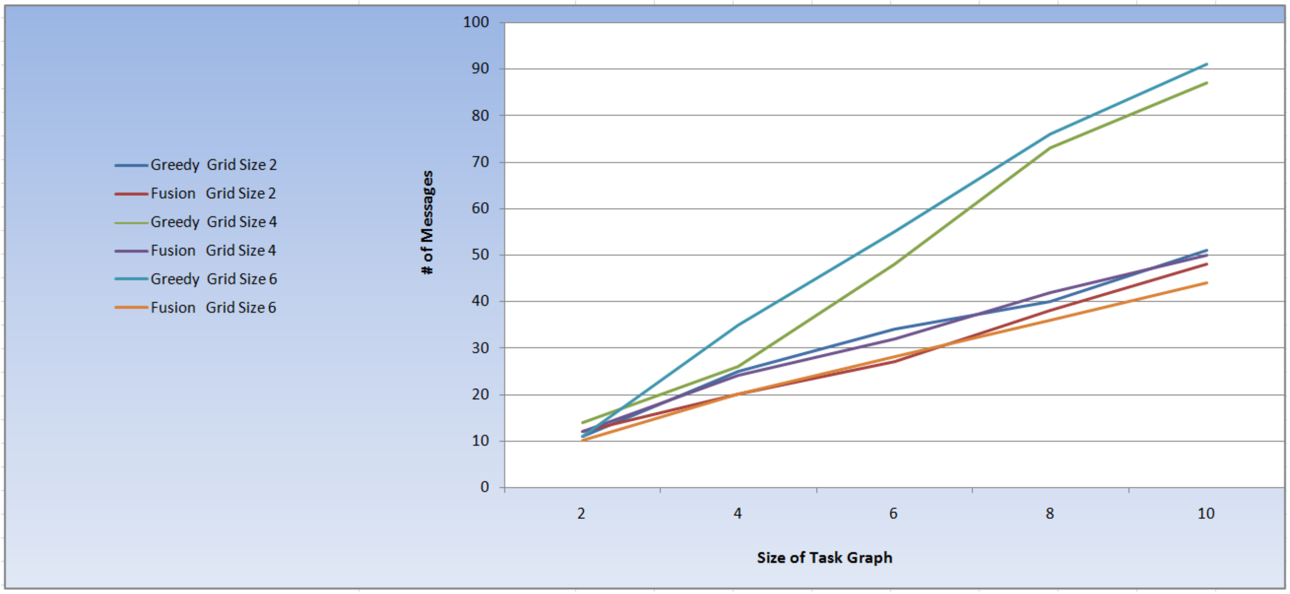


Figure 6.5 Comparing Results of Greedy and Incremental Fusion with varying Grid size

6.4. Test cases for a randomly generated topology

The results for using distributed implementation of DBSCAN and SCAN clustering techniques are given below. Fig 6.6 illustrates the effect of DBSCAN on a randomly generated sensor network topology containing 245 nodes. Fig 6.6 shows the effect of the DBSCAN clustering parameter e-neighborhood on the number of clusters and the quality of clustering. For the randomly generated topology as the E-neighborhood parameter value increases the number of clusters increases but the clustering quality indicated by number of Backbone nodes also known as the core nodes reverses after certain number of clusters indicating high fragmentation. The performance of the incremental fusion strategy implemented over network clustering is indicated by the number of messages which shows a correlation with the number of clusters. Thus clustering quality plays an important role in the quality of in-network data fusion strategy.

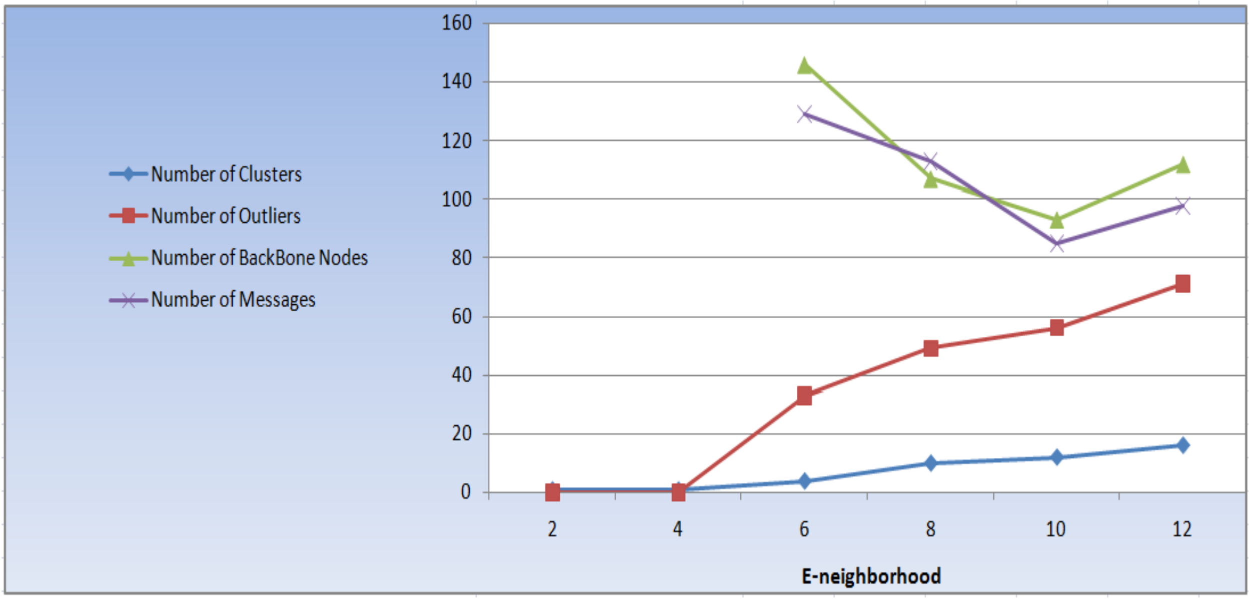


Figure 6.6 Results for Distributed DBSCAN protocol

Similarly following Fig 6.7 illustrates the effect of distributed SCAN on a randomly generated sensor network topology containing 245 nodes. Fig 6.7 shows the effect of the SCAN structure similarity score U on the number of clusters and the quality of clustering. For the randomly generated topology as the structure similarity parameter value increases the number of clusters increases but the clustering quality indicated by number of Backbone nodes also known as the core nodes reverses after certain number of clusters indicating high fragmentation. The performance of the incremental fusion strategy implemented over network clustering is indicated by the number of messages which shows a correlation with the number of clusters. Thus clustering quality plays an important role in the quality of in-network data fusion strategy.

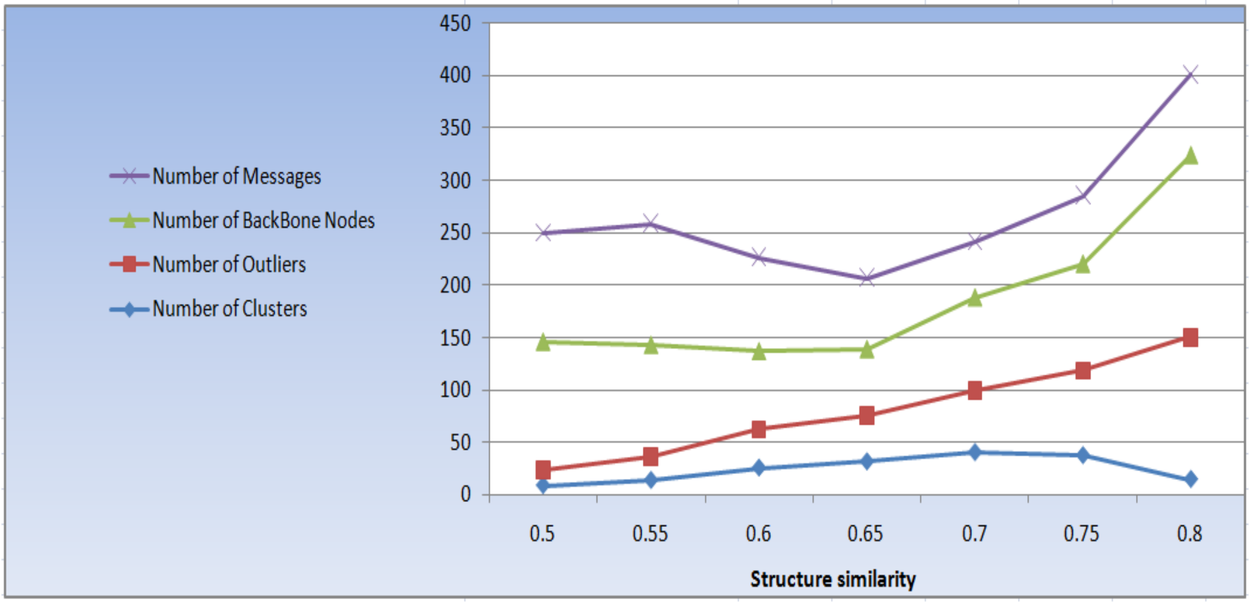


Figure 6.7 Results for Distributed SCAN protocol

The above Fig 6.7 shows a direct correlation between the number of clusters and the number of messages and the number of Backbone core nodes as a result of the clustering process. As structure similarity increases the number of clusters increases till a certain point, further increase in cluster similarity metric results in sudden increase in fragmentation which is indicated by the rapid increase number of outliers and the number of messages. Thus quality of clustering and the performance of the incremental fusion protocol is extremely sensitive to the structure similarity parameter.

CHAPTER 7 - Conclusions and future work

7.1. Conclusions

Various approaches for data fusion and aggregation to increase network life were presented, and their performance compared and discussed in this thesis .From the analysis of the results it can be inferred that efficient in-network data fusion and data aggregation can reduce the amount of communication in the network and optimise the network lifetime. When the topology of the network is known heuristics can play a significant role in reducing the network traffic by providing efficient routing protocols. When the topology is not known the network can be logically divided in the form of clusters and in-network computation can be achieved by utilising areas of high density and structural similarity called clusters. Clustering can help with routing by connecting the core nodes as the Backbone nodes and using these nodes as an overlay for intra-network and inter-network routing.

It can also be concluded that design of a good grid alignment can result in decreasing the hop-count of the system.

The clustering algorithms are scalable and a distributed implementation of the clustering algorithm terminates in finite time. Good clustering can lead towards efficient heuristics for routing protocols. Structural similarity can be an efficient metric to divide a large dense network into manageable clusters.

By comparing the various results it can be concluded that aggregation and data fusion not only reduces the network traffic but also provide efficient mechanisms for constructing shared data paths over a Backbone.

7.2. Future work

Other metrics like the latent energy levels, average response time etc. that reflect the effective life time of the network must be considered to measure the performance of the protocols.

Probabilistically selecting cluster heads and Backbone nodes based on network characteristics to reduce the total communication cost.

The role mapping procedure in the presence of dynamically changing sensor network topologies must be investigated. In order to work in an environment where node characteristics change constantly the Fusion placement must adapt to the changing network topology. This can be done by introducing a cost function that takes the network changes into consideration.

References

1 I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. “Wireless sensor networks: a survey.“Computer Networks, 2002.

2 W.R. Heinzelman, J. Kulik, and H. Balakrishnan “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks,” MobiCom 1999.

3 Bhaskar Krishnamachari, Deborah Estrin, Stephen Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," pp.575, 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02), 2002.

4 J. Reich J. Liu and F. Zhao, “Collaborative in-network processing for target tracking,”EURASIP, 2002.

5N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, “A wireless sensor network for structural monitoring,” SenSys, 2004.

6A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, “Wireless sensor networks for habitat monitoring,” WSNA, 2002.

7 Govindan, R. 2004. Data-centric routing and storage in sensor networks. In Wireless Sensor Networks, C. S. Raghavendra, K. M. Sivalingam, and T. Znati, Eds. Kluwer Academic Publishers, Norwell, MA, 185-205.

8 F. K. Hwang, D. S. Richaxds, and P. Winter. The Steiner Tree Problem, North-Holland, 1992.

9 M.R. Garey, D. S. Johnson. '~The Rectilinear Steiner Problem is NP-Complete', SIAM J. Appl. Math., 32, 826-834, 1977.

10 Jason L. Hill , David E. Culler, Mica: A Wireless Platform for Deeply Embedded Networks, IEEE Micro, v.22 n.6, p.12-24, November 2002

11 D. E. Culler, J. Hill, P. Buonadonna, R. Szewczyk, A.Woo, “A Network-Centric Approach to Embedded Software for Tiny Devices”,Proc. Int. Workshop Embedded Software (EMSOFT), pp. 114-130, Oct. 2001.

12 David Gay , Philip Levis , Robert von Behren , Matt Welsh , Eric Brewer , David Culler, The nesClanguage: A holistic approach to networked embedded systems, Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, June 09-11, 2003, San Diego, California, USA

13 NS-2, http://www.isi.edu/nsnam/ns/, 2004.

14 OPNET Technologies, Inc., “OPNET Modeler”, http://www.opnet.com/products/modeler/home.html, 2004.

15 P. Levis, N. Lee, M. Welsh, D. Culler, “TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications”, Proc. ACM Int. Conf. Embedded Networked Sensor Systems (ACM SenSys), pp. 126-137, Nov. 2003.

16 C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, “Directed Diffusion for Wireless Sensor Networking”,IEEE/ACM Trans. Networking, vol. 11, no. 1, pp. 2-16, Feb. 2003.

17 W. Heinzelman, J. Kulik, H. Balakrishnan, “Adaptive Protocols for Information Dissemination inWireless Sensor Networks”, Proc.ACM Int. Conf. Mobile Computing and Networking (ACM/IEEE Mobicom), pp. 174-185, Aug. 1999.

18 S. Lindsey, C. Raghavendra, K. M. Sivalingam, “Data Gathering Algorithms in Sensor Networks using Energy Metrics”, IEEETrans. Parallel and Distributed Systems, vol. 13, no. 9, pp. 924-935, Sept. 2002.

19 H. O. Tan, I. Korpeoglu, “Power Efficient Data Gatherting and Aggregation in Wireless Sensor Networks”, Proc. ACM Int. Conf.Management of Data (ACM SIGMOD), vol. 32, no. 4, pp. 66-71, Dec. 2003.

20 A. Manjeshwar, Q-A. Zeng, D. P. Agarwal, “An Analytical Model for Information Retrieval in Wireless Sensor Networks using Enhanced APTEEN Protocol”, IEEE Trans. Parallel and Distributed Systems, vol. 13, no. 12, pp. 1290-1302, Dec. 2002.

21 Kaufman, L. and Rousseeuw, P. J. Finding Groups in Data: An Introduction to Cluster Analysis. New York: Wiley, 1990.

22 ICHINO, M. AND YAGUCHI, H. 1994. Generalized Minkowski metrics for mixed feature-type data analysis. IEEE Trans. Syst. Man Cybern. 24, 698–708.

23 V. Batagelj andM. Bren, “Comparing resemblance measures,”in Proc. International Meeting on Distance Analysis (DISTANCIA’92), Rennes, France, June 1992.

24 MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.1. University of California Press. pp. 281–297.

25 Sergios Theodoridis & Konstantinos Koutroumbas (2006). Pattern Recognition 3rd ed.. pp. 635.

26 Abbasi, A. A.; Younis, M. A survey on clustering algorithms for wireless sensor networks. Comput.Commun. 2007, 30(14-15), 2826–2841.

27 D.J. Baker, A. Ephremides, The architectural organization of a mobile radio network via a distributed algorithm, IEEE Transactions on Communications, COM-29 (11) (1981) 1694–1701.

28 C.R. Lin, M. Gerla, Adaptive clustering for mobile wireless networks,IEEE Journal on Selected Areas Communications 15 (7) (1997) 1265–1275.

29 R. Nagpal, D. Coore, An algorithm for group formation in an amorphous computer, in: Proceedings of the 10th International Conference on Parallel and Distributed Systems (PDCS'98), Las Vegas, NV, October 1998.

30 S. Banerjee, S. Khuller, A clustering scheme for hierarchical control in multi-hop wireless networks, in: Proceedings of 20th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01), Anchorage, AK, April 2001.

31 W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks",Proc. IEEE Int. Conf. System Sciences, vol. 8, pp. 8020, Jan. 2000.

32 W. Heinzelman, "Application-Specific Protocol Architectures for Wireless Networks", PhD thesis, Massachusetts Inst. of Technology,June 2000.

[33] O. Younis, S. Fahmy, HEED: A Hybrid, Energy-Efficient, Distributed clustering approach for Ad Hoc sensor networks, IEEE Transactions on Mobile Computing 3 (4) (2004) 366–379.

[34] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". in Evangelos Simoudis, Jiawei Han, Usama M. Fayyad. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press.

[35] Xiaowei Xu , Nurcan Yuruk , Zhidan Feng , Thomas A. J. Schweiger, SCAN: a structural clustering algorithm for networks, Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, August 12-15, 2007, San Jose, California, USA