ENERGY EFFICIENT WIRELESS SENSOR NETWORK CLUSTERING ALGORITHMS AND THEIR REAL LIFE PERFORMANCE EVALUATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET ERHAN UYAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2012

Approval of the Graduate School of Informatics

Prof. Dr. Nazife Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Yasemin Yardımcı Çetin
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Altan Koçyiğit	Assist. Prof. Dr. P. Erhan Eren
Co-Supervisor	Supervisor

Examining Committee Members

Prof. Dr. Nazife Baykal	(METU, II)	
Assist. Prof. Dr. P. Erhan Eren	(METU, II)	
Assoc. Prof. Dr. Altan Koçyiğit	(METU, II)	
Assist. Prof. Dr. Banu Günel	(METU, II)	
Dr. Emrah Tomur	(INNOVA A.Ş)	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MEHMET ERHAN UYAR

Signature :

ABSTRACT

ENERGY EFFICIENT WIRELESS SENSOR NETWORK CLUSTERING ALGORITHMS AND THEIR REAL LIFE PERFORMANCE EVALUATION

Uyar, Mehmet Erhan

M.Sc., Department of Information Systems

Supervisor : Assist. Prof. Dr. P. Erhan Eren

Co-Supervisor : Assoc. Prof. Dr. Altan Koçyiğit

September 2012, 65 pages

Improvements in technology result in evolution of smart devices. One of such smart devices is wireless sensor nodes, which consist of a sensing board, a battery supply and a wireless antenna to transfer data. We can collect information from the environment by deploying thousands of these tiny smart devices. These devices can also be used to monitor natural habitats or used in giant machine parts for performance evolution. Energy efficient operation is an important issue for wireless sensor network design and clustering is one of the most widely used approaches for energy efficiency.

This thesis study aims to analyze the performance of clustering algorithms for wireless sensor networks. We propose five clustering algorithms and perform experiments by using real sensor hardware over different topologies to investigate energy efficiency of the clustering algorithms.

Keywords: Clustering, Wireless Sensor Network, Wireless Sensors

ÖZ

KABLOSUZ ALGILAYICILARIN ENERJİ TASARRUFLU GRUPLAMA ALGORİTMALARI VE BU ALGILAYICILARIN GERÇEK PERFORMANS DEĞERLENDİRMESİ

Uyar, Mehmet Erhan
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi : Yard. Doç. Dr. P. Erhan Eren
Ortak Tez Yöneticisi : Doç. Dr. Altan Koçyiğit

Eylül 2012, 65 sayfa

Teknolojideki ilerlemeler akıllı cihazların ortaya çıkmasına neden olmaktadır. Ortaya çıkan bu akıllı cihazlardan bir tanesi algılayıcı kart, güç kaynağı ve veri aktarımı için kablosuz antenden oluşan kablosuz algılayıcılardır. Bu küçük akıllı cihazlardan binlercesini etrafımıza yayarak çevremizden bilgi toplayabiliriz. Bu kablosuz algılayıcı cihazlar aynı zamanda doğal yaşamı izlemek için ya da büyük cihazların performanslarını ölçmek amacı ile de kullanılabilirler. Enerji tasarruflu çalışma kablosuz algılayıcı cihazlar için önemli bir özelliktir ve gruplama da enerji tasarrufu sağlamak için kullanılan en yaygın yaklaşımlardan bir tanesidir.

Bu tez çalışmasının amacı, kablosuz algılayıcıların gruplama algoritmalarının performansını analiz etmektir. Bu bağlamda farklı yerleşim şemalarında gruplama algoritmalarının enerji tasarruflarını incelemek amacıyla beş gruplama algoritması önerdik ve bu algoritmaların performanslarını gerçek algılayıcı cihazlar kullanarak test ettik.

Anahtar Kelimeler: Gruplama, Kablosuz Sensör Ağlar, Kablosuz Algılayıcılar

to my family...

ACKNOWLEDGMENTS

First of all I would like to thank my supervisor Assist. Prof. Dr. P. Erhan Eren and my co-supervisor Assoc. Prof. Dr. Altan Koçyiğit for being great advisors throughout my thesis study.

I also would like to thank Dr. Emrah Tomur for his great ideas, help and encouragement throughout my thesis study.

My greatest thanks to my destiny friend Gökçen Yılmaz for her endless support and help under all circumstances. I can't express my feelings with words, thank you so much for being there for me in every step of my life.

I would like to thank my friends Onur Ozan Koçak, Serhat Peker, Özge Gürbüz and Davut Çavdar for their great friendships, help and guidance during this thesis. They were very supportive in every step of my thesis.

I especially thank to my brothers Ahmet Gökhan Uyar and Ali Serhan Uyar for their great help. They waited long hours with me during experiments.

I would also like to thank my esteemed colleagues Feride Erdal, Ulaş Canatalı, Murat Ulubay, N. Nilgün Öner Tangör, my precious roommate Gül Açan Çalışır, and many more who helped and supported me. I am very lucky to work among you.

My sincere thanks to Prof. Dr. Nazife Baykal, and Assist. Prof. Dr. Banu Günel for reviewing my thesis and accepting to be on my thesis commmittee.

My ultimate thanks and appreciation to my family for their endless support and faith in me. They are always by my side, whenever I need them, thank you so much.

Finally, I want to thank to The Scientific and Technical Research Council of Turkey (TÜBİTAK) for supporting me with scholarship during my MSc study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

TDMA	Time Division Multiple Access
CDMA	Code Division Multiple Access
LEACH	Low Energy Adaptive Clustering Hierarchy
LEACH-C	Low Energy Adaptive Clustering Hierarchy - Centralized
MTE	Minimum-Energy Routing
RTS	Request to send
CTS	Clear to send
GC	General Clustering
HEED	Hybrid Energy-Efficient Distributed
ACE	Algorithm for Cluster Establishment
GPS	Global Positioning System
CH	Cluster head
TTL	Time to live
SHC	Simple Hierarchical Clustering
HHC	Hierarchical Hop-ahead Clustering
DD	Direct Diffusion
PEQ	Periodic, Event-driven and Query-based
HPEQ	Hierarchical Periodic, Event-driven and Query-based
AODV	Ad hoc On-Demand Distance Vector Routing
DCA	Distributed Clustering Algorithm
WCA	Weighted Clustering Algorithm
PR	Power Register

CHAPTER 1

INTRODUCTION

Recent improvements in technology provide us cheap and tiny electronic devices with various sensors on it. These tiny devices are called 'sensor nodes' and they have great abilities. The aim of using sensor nodes is to sense the environment and process and/or transfer collected information to an analysis center. Sensor nodes are usually battery powered and their transmission range is very low. Therefore, these sensor nodes can establish a network to propagate their data to long distances.

Wireless sensor networks idea is envisioned and defined as self-deployed, error prone, long living inexpensive communication devices that are densely deployed to collect data from physical space [1]. Another definition of wireless sensor networks is "a large-scale, ad hoc, multihop, unpartitioned network of largely homogeneous, tiny, resource-constrained, mostly immobile sensor nodes that would be randomly deployed in the area of interest"[2].

Sensor nodes can be placed regularly or they can be randomly deployed with the help of a plane, simply throwing them from air to inaccessible areas like mountains or forests. These sensor nodes may not be very powerful but they are actually very smart devices. They can establish a sensor network by self-organizing themselves, and they can immediately start transferring data packets as soon as they sense data from their coverage area.

These sensing devices are capable of sensing temperature, humidity, visual, acoustic, location and many more. There are several fields benefiting from sensor networks [1], such as military applications for border control and surveillance, environmental applications [3] for forest fire detection [4, 5], health applications for patient monitoring [6, 7], home automation and smart homes, and many other fields like agriculture [8], vehicle tracking, inventory management,

seismic activity[9] etc.

Since wireless sensor network is a new field in the literature, there are many challenges of using them. Most important challenge is energy consumption. Since these devices are deployed on unattended wide areas, replacing their batteries is not very feasible. With a pair of AA batteries a sensor node can last for 100-120h [10]. Therefore designing an algorithm with a good energy consumption mechanism is very important.

There are several ways to reduce energy consumption in sensor networks. Clustering the network, using sleep/listen cycles, using data aggregation methods, and using data propagation methods are some of them. The objective of this thesis study is to develop energy efficient clustering algorithms to partition a network into several groups for energy efficient operation.

There are many proposed algorithms and approaches for clustering to achieve energy efficient communication. However, most of these proposed studies have been done with the help of simulations. In these simulations calculations are forming the basis, like energy consumption per data packet, idle listening power consumption, or transmission range etc. Also every possibility of events has a probability of occurrence, like data losses, collisions, etc. However in real life situations we cannot calculate every possibility. Therefore performing experiments with actual sensors becomes very important.

In this study we aim to experience how sensor nodes act in normal environment. We developed several clustering algorithms employing different cluster head selection mechanisms, such as energy level or neighbor count. We performed experiments on different network topologies to observe the effect of node placement on clustering algorithms' performance. We compare each algorithm with others in terms of total transmitted and received packet counts, total coverage percentage, total number of cluster heads and member nodes, and total energy consumption.

While conducting our experiments by using actual sensors, we have encountered several problems. First of all, we have to consider that each sensor nodes' quality may be different than others. Since these devices are prototypes, their radios might not work 100% efficiently all the time. Moreover, some nodes might have hardware failures or low performances. Also testing environment may include interference, like other radio frequencies stronger than our sensor nodes. Furthermore since these sensors are powered with two AA type batteries, we

should also consider that each battery pair might have a different energy level, which effects transmission range. In order to cope with these problems we repeated the experiments several times and we present the results free of such problems as much as possible.

Due to the changing battery levels, some sensor nodes can suppress other sensor nodes' transmission signals throughout the experiments. As a result some sensor nodes might not be a part of a cluster at all.

The remainder of the thesis is organized as follows;

Chapter 2 includes definitions and the literature review on the related study. We include clustering, semantic clustering and query based clustering approaches.

Chapter 3 presents the clustering approach and proposed clustering algorithms.

In Chapter 4, we state experiment setup and present the results of experiments performed. Then, we compare the performance of proposed clustering algorithms.

Chapter 5 summarizes results obtained in this thesis study and states potential future research topics.

CHAPTER 2

LITERATURE REVIEW

In this section detailed analysis of proposed clustering algorithms, their approaches, and assumptions as well as the performance of proposed algorithms, mostly in simulation, are discussed.

A good clustering approach should have features such as;

- Fault tolerant for node failures,
- Reliable for communication between nodes,
- Low latency for delays,
- Fast reconfiguration for new paths,
- Energy saving for network longevity.

2.1 What is clustering?

Clustering is an approach to reduce communication between nodes and therefore minimize energy consumption. Researches show that clustering improves total performance of wireless sensor networks, such as communication overhead and network longevity etc.

Clustering hundreds of nodes into many controllable smaller groups may eventually increase the performance of the network. Partitioning a network into many clusters will reduce the amount of traffic and the amount of energy consumed in network [11]. Since energy efficiency is very important for network lifetime, clustering becomes crucial. If a sensor node needs to reduce its energy consumption, it should send its data packets to cluster heads firstly, instead of sending them directly to the sink [10]. As stated in [12] sensor nodes could be grouped together based on their energy levels, sensed data types, proximity to each other or many other

parameters.

There are many efficient proposed ways for selecting cluster heads. However, as a starting point of the selection process, there should be two basic criteria; (1) nodes should have a unique identifier and (2) these identifiers should be uniformly distributed among nodes [13]. Some of the methods used for cluster head selection are; choosing nodes which are closer to the base station, choosing nodes randomly, or choosing the nodes that have highest or lowest parameters than neighbors, in which parameters could be residual energy level, neighbor count, package count, sensed value, unique identifiers etc.

However, using simple clustering algorithms is not always efficient. If simple clustering algorithms such as selecting nodes with lowest or highest identifiers are applied, same nodes will be selected as cluster head many times. This results in quick energy drain of these selected nodes. Therefore, selection of cluster heads is also crucial to distribute load evenly among other nodes in order to minimize energy consumption.

[14] lists some of the reasons for using clustering in wireless networks such as; to perform data aggregation in order to reduce total energy consumption and reduce the total number of packets transmitted, to disseminate queries to members, or to form an effective routing algorithm for the network. Also, clustering can be performed in single-level, which is the mostly used approach, or multi-level clusters can be performed - which is creating clusters inside a cluster.

Data aggregation is collecting data from member nodes, and transmitting the final data in a single packet to sink node. Data aggregation is widely used in clustering approach, because data from member nodes are collected by cluster heads and sent to the sink in a single packet, in order to reduce network traffic. When a sensor node receives two packets from two different source nodes, it can process incoming data packets and calculate the average readings, in order to send the final value as a single data packet. Another choice for a sensor node to aggregate data is to merge two different readings into same packet and sending the final packet to its destination. Both methods will reduce the energy consumption and network data traffic [15].

Data aggregation models are necessary to avoid redundant data packets, which creates too much traffic, and to minimize energy consumption [16]. Besides, controlling all members in the network is easier when they are controlled as a group. In case of a query based approach,

data aggregation algorithms work in the opposite direction to disseminate data query to members, in order to change event thresholds they store or in order to collect different data from network.

Data propagation techniques are also very effective in controlling energy consumption. There are two ways of sending a packet to the sink node. Firstly, node can decide to send its packet directly to the sink, in a single hop fashion, which requires more energy. Secondly, a node can choose one of its neighbors to relay its packets to the sink, in a multi hop fashion. If the second solution is used, energy consumption will be smaller, because the distance is smaller between two nodes. Energy consumption increases incrementally when the distance between a node and sink is increased. [17]

Data collection can be performed in a periodic fashion, where nodes send their sensed data in defined intervals, or event driven fashion, where nodes send their sensed data only when there is a significant event to sense and alert the sink.

Centralized approach and distributed approach are the most commonly used approaches to form clusters. Distributed approach is more common in large scale networks, because centralized approaches require knowledge of the network topology and that is time and energy consuming [13].

Distributed approach [13] is more useful for our proposed algorithms, because our proposed algorithms are scalable to large scale networks. Also nodes decide whether they will become a cluster head of a group, or one of the members of a group, based on the information received from its neighbors and on many criteria coded in our proposed algorithms.

Two alternative ways to form clusters and cluster tree is top-down approach [18] and bottom-up approach [19]. In top-down approach root selects its neighbors and they become cluster heads, then they form their own clusters. This approach provides more control in forming clusters and cluster tree. However bottom-up approach forms individual clusters and later try to gather them together. This increases the communication overhead between nodes.

2.2 Clustering Algorithms

In 1981, Baker et al. [20] proposed a linked clustering algorithm which is a self-starting and distributed clustering algorithm. Proposed algorithm has two phases, formation of clusters and linking clusters. There are three types of nodes; ordinary nodes, cluster head nodes, and gateway nodes which links two cluster heads, and also construct the backbone of the network. They performed simulations with different network sizes to collect performance results. In each run, according to the steps of the algorithm some nodes become cluster heads, while some remain ordinary nodes. Some cluster heads degraded to ordinary nodes as their coverage area is covered by another cluster head and overlapping occurs. This basic clustering protocol is the simplest step of clustering approach.

In 2000 Heinzelman et al. proposed ”an application-specific” low energy consuming clustering scheme (LEACH) [21][22] with the goal of increasing network lifetime. In this algorithm, randomly selected nodes acts as ”local base-stations” to a group of regular nodes. Main elements of the proposed algorithm can be stated as: 1) Load balancing via randomly selected nodes in each round to minimize energy consumption. 2) Local data processing in order to reduce communication load. Simulation results showed that LEACH performs better than LEACH-C [22], minimum-energy routing (MTE) [23] and static clustering. Metrics used in the simulation are network lifetime and amount of data transferred. Results show that nodes can communicate longer until first node dies when using LEACH algorithm. Moreover, rotating the cluster head role will reduce the energy consumption when compared to fixed role.

In [24], Lindsey et al. proposed an energy efficient communication protocol called PEGASIS. Authors compared their near-optimal chain based protocol with LEACH protocol proposed in [21][22]. Authors performed simulations to complete their study, and results showed that 100 to 200 percent improvement achieved as regards network lifetime.

In 2002, Handy et al. [25] extended Heinzelman’s LEACH protocol in [21][22]. Aim of this extension was to reduce energy consumption of the network. Authors explained their simulation results according to two metrics, ”First Node Dies” and ”Half Node Dies”. Results showed that 30% and 21% improvements respectively against LEACH, with the proposed modifications.

Basagni in [26] described two approaches for the phases of clustering algorithm. First ap-

proach is, Distributed Clustering Algorithm, proposed for the set-up based and assumes that there is no mobility. Second approach is, Distributed Mobility Adaptive Clustering, proposed for set-up/maintenance phase and handles mobility of nodes. In both approaches, cluster head selection is based on a weight given to sensor nodes. The only requirement in the given algorithms is the knowledge of one-hop topology. As a result for the first approach, better cluster head selection is achieved and second approach adds mobility features to previously proposed algorithms.

In 2001, Banarjee et al. [27] proposed a multi hop clustering algorithm for large number of nodes, to maintain and organize these nodes into clusters. Authors listed their desired design goals as; all nodes are part of some cluster and all clusters are connected, all nodes should have similar transmission range, and all clusters should be same size, etc. The algorithm works by finding a spanning tree, and has two phases, cluster creation and cluster maintenance. Cluster creation consists of discovering tree and formatting clusters. Cluster maintenance is performed when cluster quality drops below a threshold, when new nodes join, or existing nodes die. Authors simulated their algorithm with 700 to 1100 nodes and presume that no RTS-CTS messages used. Therefore, they overcome package lost or collisions by soft-state timeouts. Results showed that connectivity is adversely proportional to diameter of a cluster. When node connectivity is increased cluster diameter will decrease.

Younis et al. [28] proposed an approach for energy-efficient clustering of nodes. Their approach is based on residual energy and an additional parameter of node, which can be node degree, node id or any other parameter. Their "Hybrid Energy-Efficient Distributed (HEED)" clustering algorithm does not make any assumptions about network. One important aspect of the proposed approach is every node can be both member and cluster head. Authors compared their algorithm with general clustering (GC) protocol (DCA [26] and WCA [29]), which uses only residual energy for cluster head election. Results showed that GC can guarantee that nodes with higher residual energy will become cluster heads, and cluster heads selected by HEED has lower average residual energies than nodes selected by GC. Besides this negative effect, HEED can produce more balanced clusters and cover most of the network. Authors simulated and compared their algorithm with an improved version of LEACH (gen-LEACH) algorithm in [21]. Simulation results showed that HEED uses less energy when clustering nodes than gen-LEACH, but original LEACH uses lesser energy than both with its assumptions. HEED prolongs network lifetime and propose desirable clustering characteristics such

as balanced network, and higher non-single-node clusters.

In 2004, Chan et al. [14] proposed a new emergent node clustering algorithm called "Algorithm for Cluster Establishment (ACE)", which has great efficiency in clustering nodes. Their algorithm does not use any localization information. The proposed algorithm has two main phases; cluster formation and mitigation. For the formation phase, each node can only chose one cluster head but it can follow multiple cluster heads until choosing one of them. Mitigation only occurs when two clusters overlap to minimize overlapping area. After the first iteration, clusters mitigate to non-overlapping spaces. Simulations are performed to compare the performance of ACE algorithm. The Node ID [20] and the Node Degree [30] algorithms are used for comparison. In Node ID algorithm, node with the highest ID among neighbors will become the next cluster head. In Node Degree algorithm, node with the highest degree (one of many parameters changing from algorithm to algorithm, which can be energy levels, neighbor count or distance to Sink node etc.) will become the next cluster head. Simulations were performed in highly dense environment with 2500 nodes. Results showed that ACE has the lowest average cluster sizes. Even with some packet loss rate, ACE is still superior than Node ID and Node Degree algorithms. In three rounds ACE forms a highly efficient coverage over the network.

Lin et al. [30] proposed a new self-organizing multihop clustering algorithm based on node ID. Their algorithm uses code division multiple access to prevent collisions and supports mobility and topology changes. The main goal of the proposed algorithm is to cover the entire network. Their basic assumption is that every node has a unique ID and every node knows its neighbors IDs. They used transmitter-based code assignment, which means that every node using same transmitting code within a cluster, to prevent inter-cluster collisions. Added to that, they also implement TDMA schemes to prevent intra-cluster collisions, where each node has a different time slot to transmit. Simulations are performed to compare the performance of the proposed algorithm. Proposed Adaptive Clustering algorithm is compared with PRNET [31], MACA/PR [32] and Cluster TDMA [33] algorithms. Results showed that proposed algorithm has lower packet loss and has more packet transmission rate, but it performs similar to Cluster TDMA algorithm which has a difficult implementation.

Gupta et al. [34] proposed a load balanced clustering algorithm for sensor networks. They aim to distribute node balance among many clusters to minimize heavily loaded clusters. Network

consists of sensors and gateway nodes, and they assume that all gateway nodes are within communication range with each other. As a first step, nodes send their packets according to TDMA schedule to prevent collisions. Proposed algorithm has two phases. In the first phase, gateway nodes discover neighbor nodes, and in the second phase, gateways form clusters based on communication cost of each node. Simulations are performed on collision free and no-packet drop network. Results were compared with shortest distance clustering algorithm. They showed that clusters are uniformly distributed when using the proposed algorithm. Also, performance of the proposed algorithm remains constant in high density networks. However, average energy consumption is higher than shortest distance algorithm when the number of clusters is small.

2.3 Semantic Clustering

In 2010 Liu et al. [35] proposed an energy efficient data gathering scheme for heterogeneous networks called EDGA, to achieve less energy consumption and therefore, to increase network lifetime. Authors defined their network model with several assumptions as follows: network is densely deployed and nodes do not move, all nodes should be time synchronized, there is only one base station to listen, nodes are not aware of their locations, nodes can change their transmission power, and lastly, all nodes have different starting energies. EDGA algorithm elects cluster heads according to a node's election probabilities, which is calculated by using remaining energy of a node. EDGA elects optimal number of cluster heads to maintain lowest energy consumption. Also, algorithm defines active members without compromising network coverage to reduce communication overhead. Simulation results showed that EDGA outperforms LEACH algorithm, in the meanings of network lifetime and active node counts. Nodes die slowly when deployed under EDGA algorithm.

Nam et al. [36] proposed an adaptive cluster head selection, which re-positions the cluster head node based on its position to its member nodes. This approach aims to use energy efficiently, by reducing energy spent on communication. LEACH-C uses the similar approach by adding GPS to determine positions, but this is not feasible due to high cost and energy consumption. The proposed algorithm works by first collecting distance values of the farthest sensor node and the closest cluster head, and then cluster head subtracts two values and if the results are negative cluster head repositions itself closer to its farthest member, otherwise

positions itself closer to its nearest cluster head neighbor. Simulation results performed on 100 nodes showed that the optimal cluster size is 16-25 nodes and is achieved over 50% of the rounds by the proposed algorithm whereas in LEACH algorithm clusters often have less than ten or over thirty nodes. Distance results showed that proposed algorithm shows similar results to LEACH-C and the average distance between two cluster heads is 20.88m.

In 2006, Bouhafs et al. [37] proposed a new clustering scheme based on semantic information of the nodes. The goal of the proposed scheme is to reduce energy consumption and prolong network lifetime. In proposed approach base station starts the clustering scheme by advertising a query. Nodes that satisfy the query will start ”cluster formation phase” and nodes which do not satisfy the query will only disseminate it to their neighbors. In this approach, member nodes can have more than one cluster head; in order to build a hyper-tree rather than simple tree and to overcome node failures and coverage problems. Different from LEACH algorithm, in this approach data aggregation is done at each level of the network tree to reduce energy consumption of the cluster head spent on data aggregation. Therefore, network lifetime is longer than when data aggregation is done only by cluster head. Simulation results showed that 95% percent of the total energy consumed when 90% of the nodes are alive in proposed algorithm and 35% of the nodes are alive in LEACH algorithm. Also, LEACH algorithm sends more data messages than the proposed algorithm which consumes more energy.

In 2010, Tan et al. [38] proposed a clustering algorithm with optimal tiers to achieve energy efficiency, named as COTE. Authors first examined the role of tiers in cluster formation and then proposed a multi-tier energy efficient clustering algorithm. In each tier, to elect CHs, candidate CHs broadcasts their energy levels to be elected as CH, and when selected, they manage member nodes in their cluster. Authors proposed an optimal tier calculation based on size of the network and compared simulation results based on this calculation. For 100 nodes, optimal network optimal tier number is 7 and for 800 nodes, optimal network tier number is 3 for the maximum energy consumption. Assumptions in these calculations are; the base station is in the center, there is unlimited data to send, every node can reach its cluster head directly, and sensing environment is contention free. According to the results, proposed algorithm performs slightly better than compared algorithms.

2.4 Query Based Clustering

Bandara et al. [18] proposed a top-down approach clustering algorithm, which does not require location, neighborhood information, time synchronization, or network topology. Authors perform cluster formation based on time to live values, where TTL exceeded a new cluster is formed. First algorithm with TTL=1 is Simple Hierarchical Clustering (SHC) and second algorithm with TTL=3 is Hierarchical Hop-ahead Clustering (HHC). Simulation results showed that in SHC, clusters are less circular and they form too many clusters with different sizes. However, in HHC, clusters are more circular and they form less clusters with high uniform sizes. With appropriate parameters desirable results can be achieved such as more circular clusters, uniform cluster sizes and depths.

Boukerche et al. [39] proposed an optimal energy dissipating and fault tolerant clustering algorithm. Proposed algorithm starts with the setup phase, in which nodes with highest energy will become cluster heads. Also, each cluster head discovers nearest cluster heads and stores the path through its one-hop members. Lastly, free node discovery is performed in setup phase when free nodes broadcast their existence to their neighbors. Energy efficiency is provided by the nearest node tables. Each cluster head stores two of its one-hop members to each neighbor cluster heads to minimize network path. Fault tolerance is provided by also this nearest node tables. If a cluster head needs to transmit a packet for the second time, it uses the alternate closest member to the destination cluster head to minimize the probability of using the same faulty nodes while transmitting. Authors did not perform any simulation or an experiment; therefore, they only provided proof of correctness of the proposed methods by formulas.

Zhang et al. [40] proposed a cluster based query protocol that aims to reduce energy consumption within the sensor network. The proposed algorithm provides solutions to locating sensor, fault tolerant network operations and energy efficient data processing. Authors' uses laser beams to stimulate sensors and their one hop neighbors. They also propose a sensor cluster location algorithm to locate and elect cluster heads. They assume that the nodes are not mobile and they are deployed on open-space area. Their cluster based query protocol aims to minimize energy spent on transmitting data packets, and perform data aggregation on cluster heads. They performed several simulations on NS2 [41] simulator, and they also compared simulation results with analytical results.

In 2004, Popovski et al. [42] proposed a design for clustered network and data aggregation. Proposed design includes random cluster head selection and conflict resolution algorithm. Simulation results showed that proposed algorithm runs faster and requires less messages and time to complete. This reduces the communication overhead and therefore reduces the energy spent. Authors stated that this achievement is accomplished by localized algorithms, which performs data aggregation and cluster formation. Therefore, we can understand that clustering and local data processing is very important and saves more energy.

In 2005, Boukerche et al. [16] proposed a hierarchical cluster based approach (HPEQ) that collects and transmits collected data to sink efficiently from group of nodes. Proposed algorithm selects nodes with more energy as aggregators in order to transmit data to sink. Aggregators advertise their ID to neighbor nodes and form a cluster. After a period of time, another node is selected as aggregator to minimize energy drain of certain nodes. Each aggregator has a time-to-live parameter which allows nodes to know how far they located from their aggregator. Aggregator-to-sink packets are performed in a multi-hop fashion, based on routing tables formed during cluster formation. Simulations are performed by using NS-2 [41] network simulator and HPEQ performance is evaluated with average event delay, average event delivery ratio, and average dissipated energy. Results are compared with Direct Diffusion [43] and PEQ [44] algorithms. Results showed that with a few source nodes HPEQ's performance is similar or worse than PEQ and DD, however when source number is increased, HPEQ outperforms both algorithms in terms of evaluated criteria.

Intanagonwiwat et al. [43] proposed a data centric, energy efficient "directed diffusion" approach to collect data from network. Their proposed algorithm works by receiving interests from sink nodes and propagating interested data by using reverse path approach to sink. Intermediate nodes can also aggregate data if required. Key features of the proposed approach are; being data centric, neighbor to neighbor communication, therefore there is no need for routers and implementing reactive routing scheme which is on-demand routing similar to AODV described in [45]. Ns-2 simulator is used to compare direct diffusion with flooding and omniscient multicast. Flooding uses to much energy since it is always in broadcast mode. However since both direct diffusion and omniscient multicast using shortest path approach, direct diffusion also implements in network processing. Therefore, it minimizes its energy usage.

As seen from all above proposed algorithms many researchers proposed clustering algorithms with very different approaches to select cluster heads. Some of the approaches are random selection, selection based on weight of various factors, or selection based on parameters like energy, node ID, communication cost, position of a node, or TTL value etc. Moreover, in these algorithms data aggregation can be completed on members, cluster heads, or only in sink node.

CHAPTER 3

PROPOSED CLUSTERING ALGORITHMS

In wireless sensor networks, data collection is an essential part of the applications and there are many ways to develop an energy efficient data collection algorithm. Clustering is one of such approaches. In clustering, sensor nodes are formed as a group, and responsibilities are given to nodes such as cluster head or member nodes. Member nodes are responsible to collect data and transmit it to their cluster heads, and cluster heads are responsible to send this data to the sink. In this thesis, in order to establish an energy efficient data collection approach, we proposed 5 different clustering algorithms.

In this chapter, the algorithms and their specific cluster head selection methods are described. In section 3.1 clustering approach is explained, in section 3.2 packet types used in the algorithms are introduced, in section 3.3 sensor node types are given, in section 3.4 timers and in section 3.5 collision avoidance method are explained, in section 3.6 proposed algorithms are described and finally in section 3.7 cycles are briefly explained.

3.1 Clustering Approach

As it's explained more elaborately in section 2.1, clustering is an approach to reduce communication overhead in order to maximize energy efficiency. Clustering might also be beneficial for the reliability of the network or wider area coverage. There are basically three types of nodes in clustering approaches: Cluster heads, Member nodes and a Sink node. Member nodes send their data packets to their cluster heads and cluster heads send their data packets to the sink. In this scenario, nodes may aggregate data before sending their data to sink to reduce number of packets flowing in the network. As shown in Figure 3.1 black filled cir-

cle indicates cluster head node, gray filled circles indicate member nodes and black arrows indicate the path from cluster head to sink.

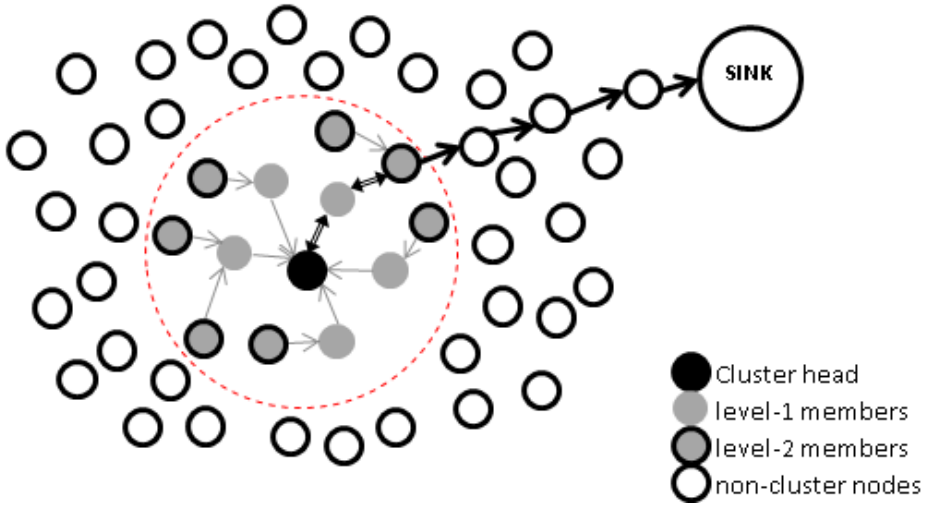


Figure 3.1: Clustering scheme when depth limit is 2

In our clustering approach, there is a depth limit value, and this limit indicates the maximum number of hops between a member and its cluster head. The depth limit can vary depending on the requirements of an application. If a cluster's depth limit is **D**, a member must be able to send its data to its cluster head at maximum **D** hops. In Figure 3.1, a sample cluster with depth limit 2 is presented. For the rest of the thesis we refer depth 1 members as level-1 members, depth 2 members as level-2 members and so on.

Another sample cluster network for depth limit 2 is shown in Figure 3.2. In this figure there are 3 clusters. Long dashed cluster on bottom right corner consists of a cluster head (CH) and 5 members, 2 of them are in level-1 (M1) and 3 of them are in level-2 (M2), dotted cluster in the middle consists of just a cluster head, and dashed cluster on upper left corner consists of a cluster head and 2 level-1 members.

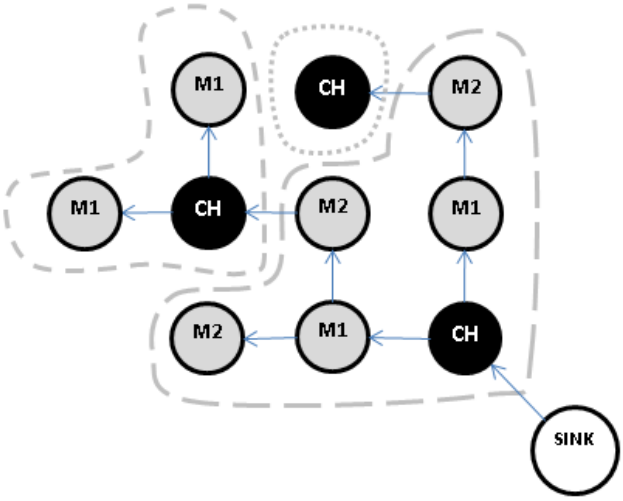


Figure 3.2: Example of clustering

3.2 Packet Types

There are four packet types used in our algorithms.

- **Initiate-cycle (IC) packets:** IC packets are broadcasted at the beginning of each cycle by the Sink node, and these packets inform nodes about the start of a new cycle. Following fields are included in IC packets:
  - Application id - indicates the cluster selection algorithm to be used.
  - Cycle id - the cycle identifier. Each cycle is assigned a unique identifier selected from an increasing sequence.
  - Sender id - indicates id of the node forwarding the packet.
  - Round count (N) - indicates the required number of data transmission rounds.
  - Round length (T) - indicates how long (seconds) a round should last.

  - Wait time (W) - a parameter used to determine how long (seconds) member nodes wait before sending the first data packet.
  - Depth limit (D) - indicates the maximum allowed levels in a cluster.
  - Energy value - indicates energy level of a node forwarding the packet.

- **Cluster-join (CJ) packets:** Cluster join packets are used to request nodes to join to an existing cluster. The fields in cluster-join packets are as follows:

  - Application id - indicates the cluster selection algorithm to be used.
  - Cycle id - the cycle identifier. Each cycle is assigned a unique identifier selected from an increasing sequence.
  - Sender id - indicates id of the node forwarding the packet.
  - Cluster id - indicates the id of the cluster head node.
  - Depth level - indicates sending node's level in the cluster.
  - Round count* - indicates the required number of data transmission rounds.
  - Round length* - indicates how long (seconds) a round should last.
  - Wait time* - a parameter used to determine how long (seconds) member nodes wait before sending the first data packet.
  - Depth limit* - indicates the maximum allowed levels in a cluster.

    (* signed fields are copied from the IC packet that leads to cluster formation.)

- **Member to Cluster head data (M2C) packets:** M2C packets are used to carry data from level-2 members to level-1 members or they carry aggregated data from level-1 members to cluster heads. A M2C packet includes following data fields;

  - Cycle id - the cycle identifier. Each cycle is assigned a unique identifier selected from an increasing sequence.
  - Sender id - indicates id of the sensor node sending the data.
  - Round number - indicates the round in the cycle.
  - Cluster id - indicates id of the connected cluster head.
  - Value - aggregated data.
  - Node count - indicates the number of aggregated values.

- **Cluster head to Sink data (C2S) packets:** C2S packets are used to carry aggregated data from cluster heads to sink. A C2S packet includes following data fields;

- Cycle id - the cycle identifier. Each cycle is assigned a unique identifier selected from an increasing sequence.
- Sender id - indicates id of the sensor node sending packet.
- Round number - indicates the round in the cycle.
- Cluster id - indicates id of the cluster head sending data.
- Value - aggregated data.
- Node count - indicates the number of aggregated values.

3.3 Sensor Node Types

In our algorithms, we used 4 types of nodes. In the beginning of each algorithm there are 2 standard node types; Sink node - with id=1, and Unassigned nodes - with node ids 2 to (N-1) where N is the number of nodes in the network. Unassigned nodes are later become member or cluster head nodes.

- Sink node starts each cycle by broadcasting first IC packet.
- Unassigned nodes are the sensor nodes not assigned to any role, they receive IC packets and broadcast them if necessary, and later they become a member or a cluster head.
- Member nodes are members in a cluster.
- Cluster head nodes are cluster heads that form clusters.

3.4 Timers

Our Algorithms use two timers, one of them is used in cluster head selection process and second one is used in rest of the processes like joining clusters, sending packets etc.

3.5 Collision Avoidance

In order to prevent collisions, we implemented a very short random waiting time between (0, S) seconds before each packet transmission. Therefore we reduced collision probabilities of packets.

3.6 Proposed Algorithms

3.6.1 Algorithm 1 - Clustering based on first responders without waiting time

In this algorithm, nodes elect themselves as cluster heads, right after they receive an IC packet. Then each cluster head broadcasts their own CJ packets to invite its neighbors to join its cluster. In this algorithm, we do not employ any intelligent cluster head election method. Nodes declare themselves as cluster heads when they received an IC packet, or they join clusters as members when they receive a CJ packet from one of their neighbors, after checking depth limit and depth level values. Flow of Algorithm 1 is described in Figure 3.3.

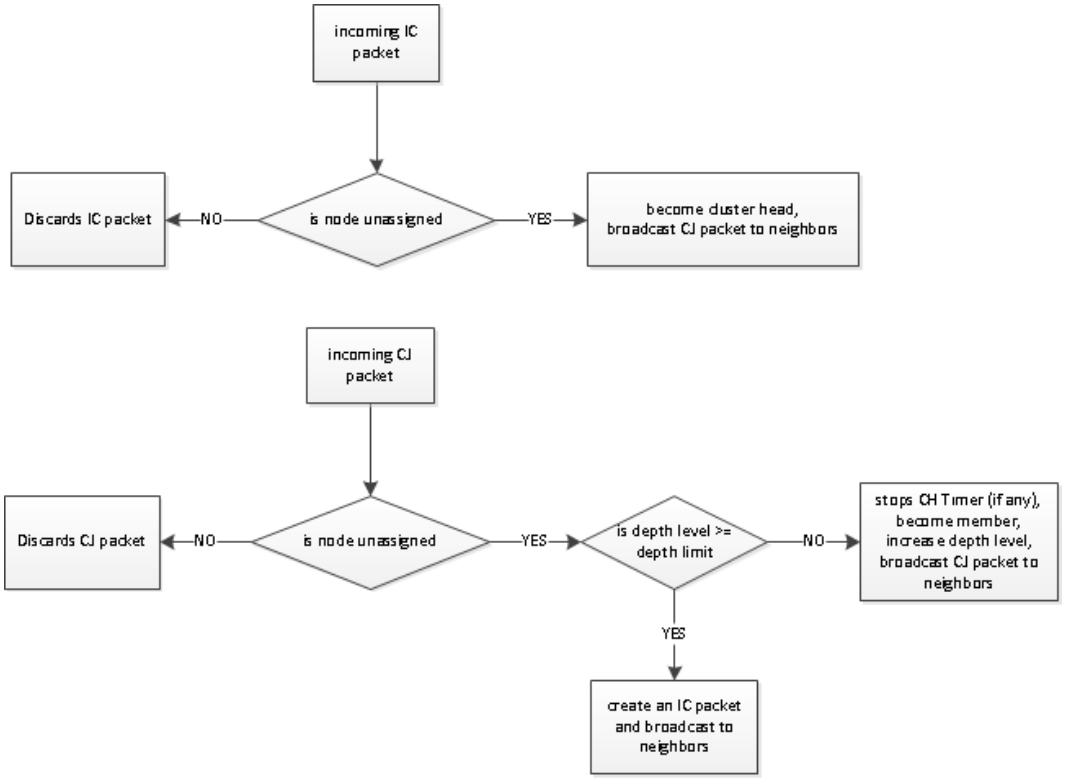


Figure 3.3: Flow chart of Algorithm 1

3.6.2 Algorithm 2 - Clustering based on first responders with random waiting time

Differently from Algorithm 1, in this algorithm nodes will wait a random time between (0, T) seconds right after they receive an IC packet to elect themselves as cluster heads. Nodes that choose the smallest waiting time values, elect themselves as cluster heads and broadcast their

CJ packets first. If a node that is still waiting to elect itself as a cluster head receives a CJ packet from another node, it cancels its cluster head selection timer and becomes a member of a cluster according to depth level and depth limit values of CJ packet. Flow of Algorithm 2 is described in Figure 3.4.

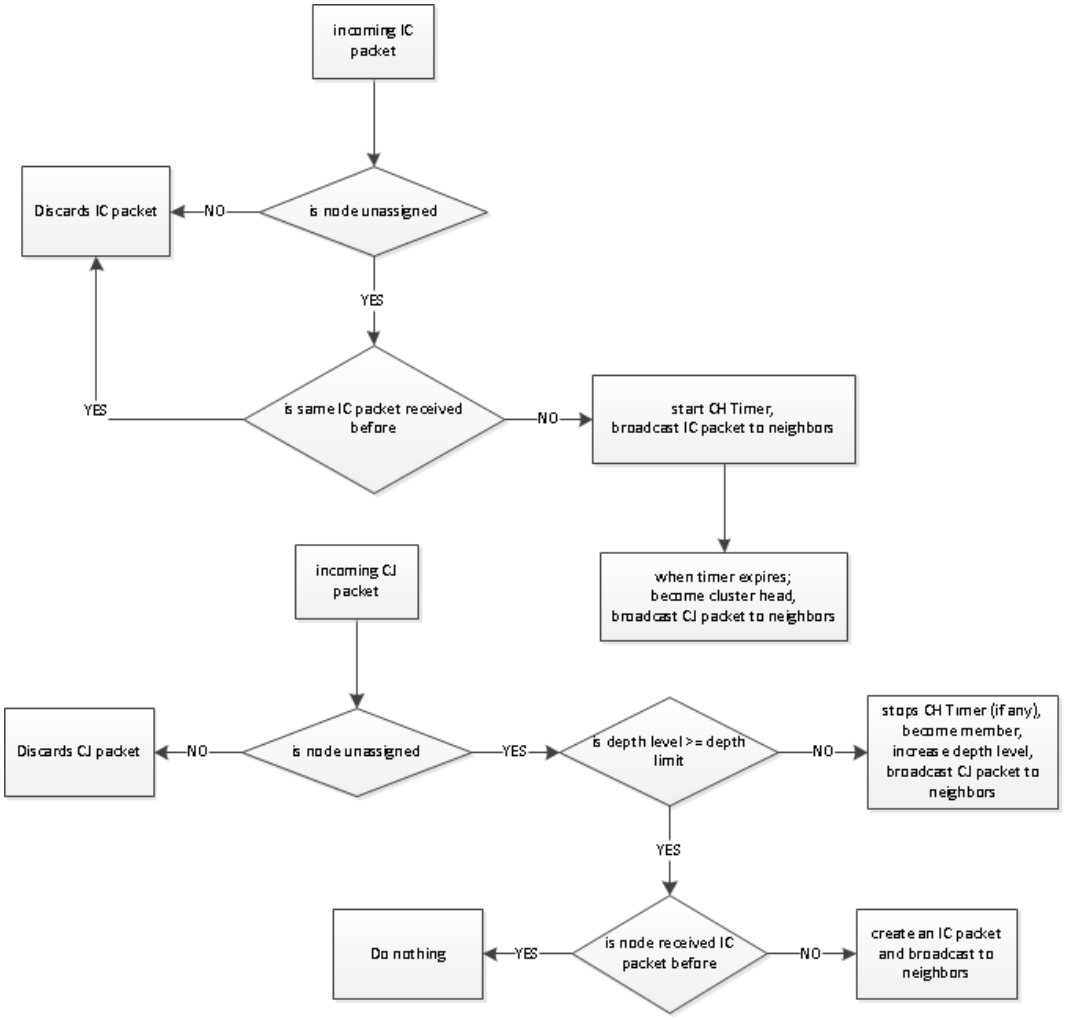


Figure 3.4: Flow chart of Algorithm 2

3.6.3 Algorithm 3 - Clustering based on highest neighbor count

In this algorithm, nodes decide who is going to be a cluster head based on their number of neighbors. During broadcast of IC packets, nodes count number of their neighbors, in order to know how many neighbors they have. If a node has **too many** neighbors, then it waits less time to elect itself as cluster head. Neighbor count of a node is inversely proportional to the

waiting time. Flow of Algorithm 3 is described in Figure 3.5.

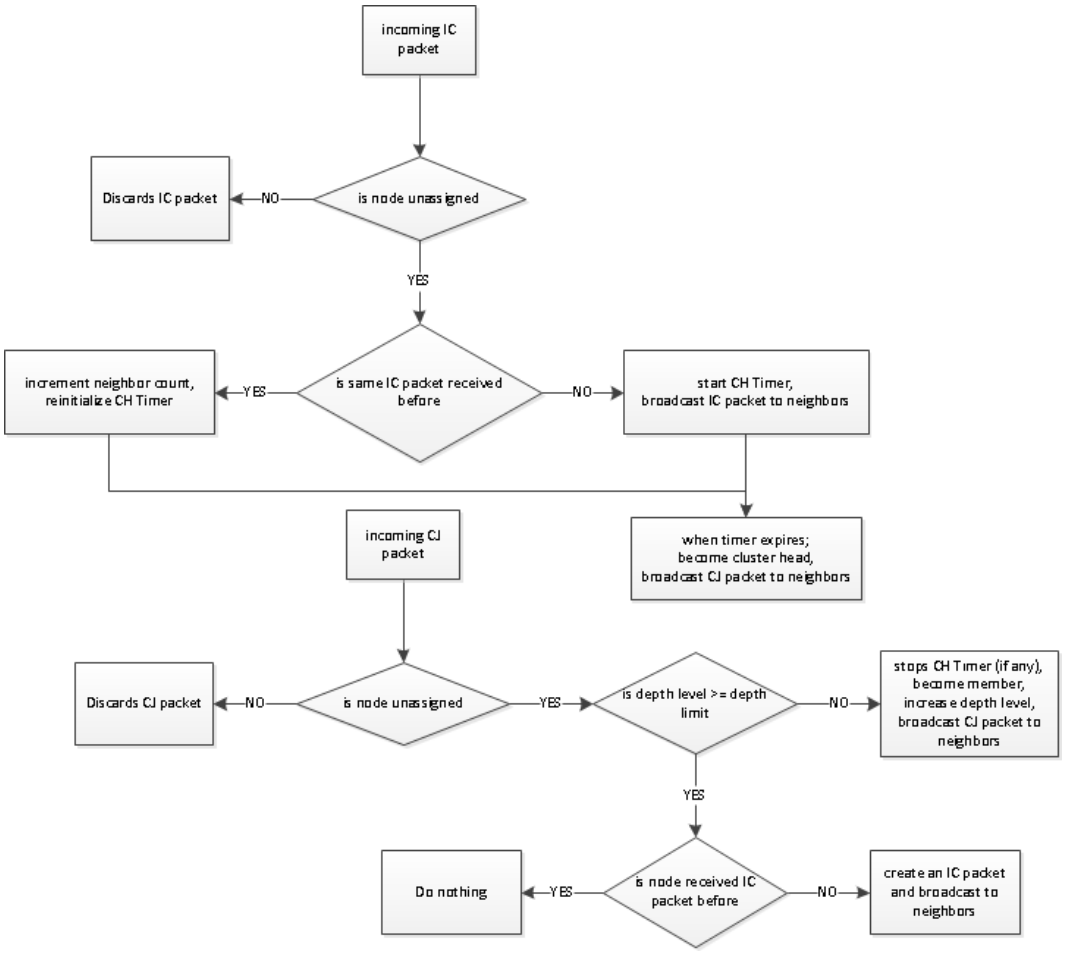


Figure 3.5: Flow chart of Algorithm 3 and Algorithm 4

3.6.4 Algorithm 4 - Clustering based on lowest neighbor count

Differently from Algorithm 3, in this algorithm neighbor count of a node is directly proportional to waiting time. If a node has **too few** neighbors, then it waits less time to elect itself as cluster head. Flow of Algorithm 4 is described in Figure 3.5.

3.6.5 Algorithm 5 - Clustering based on energy level

In this algorithm, nodes decide who is going to be a cluster head based on their energy levels. Nodes broadcast their energy levels with IC packets. If a node has the **highest** energy levels among its neighbors, it elects itself as a cluster head after a period of waiting between (2, T)

seconds. Otherwise it stops its cluster head selection timer, and waits for a CJ packet. Energy level is inversely proportional to waiting time. We calculated energy levels as given in below formula;

$$\text{Energy Level} = \frac{1}{\frac{R_X}{10} + T_X}$$

$$T_X = Transmitted\ packet\ count$$

$$R_X = Received\ packet\ count$$

Flow of Algorithm 5 is described in Figure 3.6.

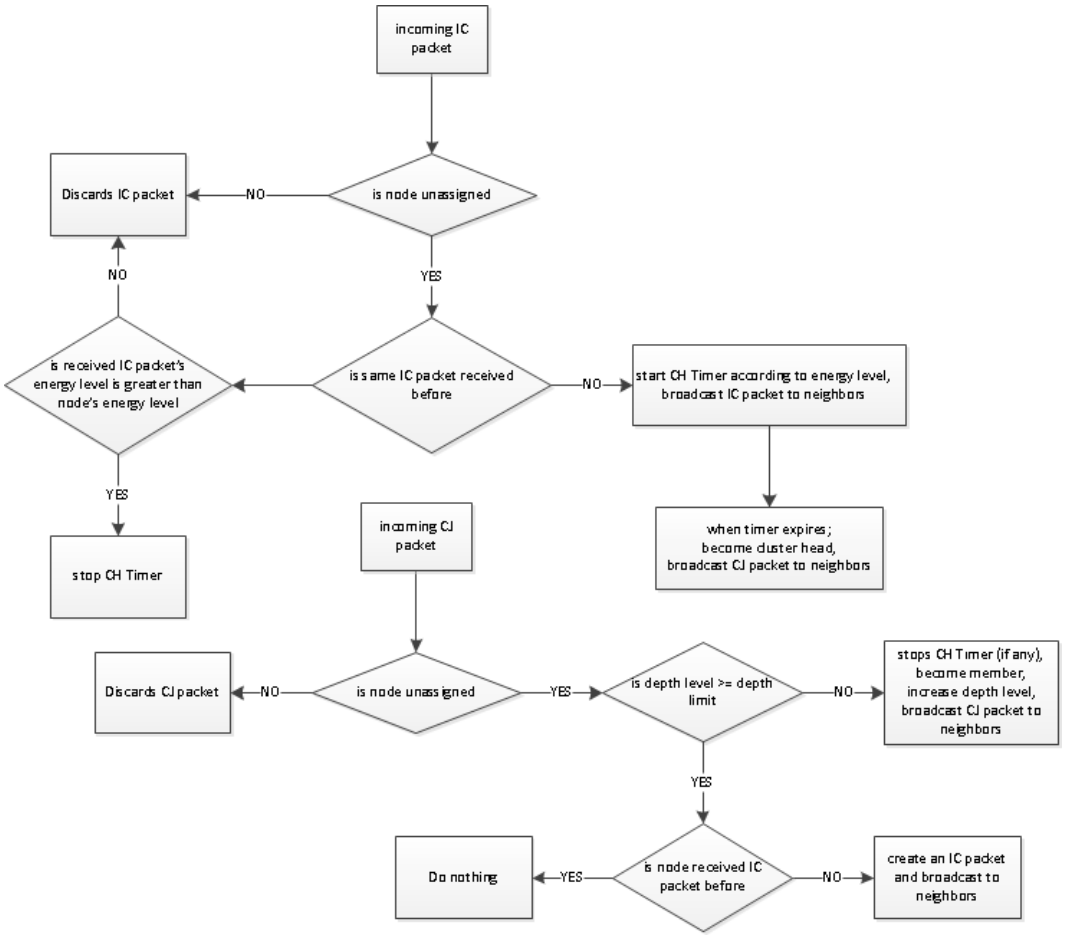


Figure 3.6: Flow chart of Algorithm 5

3.7 Cycles: Cluster formation and data transfer rounds

In each algorithm, a cluster formation and a data transfer period constitute a cycle. Each cycle starts with a cluster formation initiated by the sink node. After the clusters are formed, data transfer rounds take place, and after **N** rounds of data transfer the cycle finishes. Cycles and rounds are depicted in Figure 3.7.

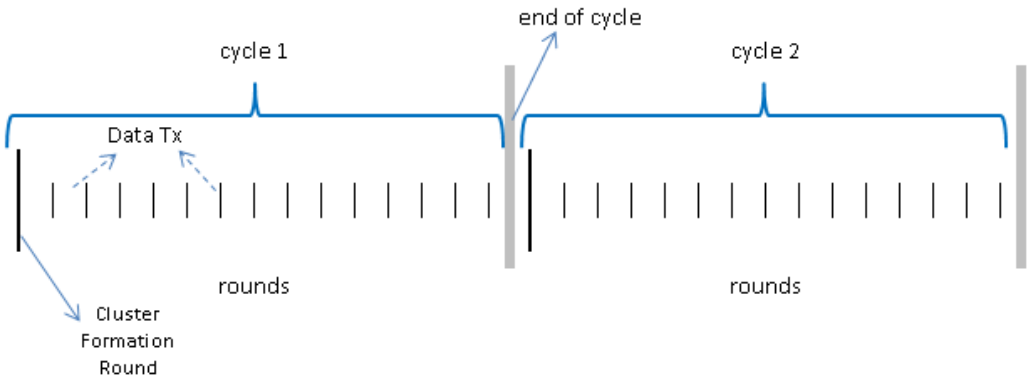


Figure 3.7: Cycles and rounds

Each cycle is initiated by the sink node by broadcasting the IC packets with a distinct cycle id. When a node receives an IC packet, first it checks the cycle id on the packet. If the receiving node hasn't received any IC packet with the cycle id greater than or equals to the received IC packet's cycle id, it processes the IC packet's content. After receiving an IC packet, the cluster formation starts.

3.7.1 Cluster Formation

In cluster formation step nodes either form a cluster or join to a cluster according to the IC or CJ packets they received. This step contains two main parts;

- Cluster head selection: This part starts with receiving an IC packet with a new cycle id. Each node broadcasts the IC packet they received only once, but nodes process each IC packet they receive. When processing IC packets, depending on the type of the algorithm, each node determines the priority of electing itself as a cluster head. If a node decides to elect itself as cluster head, then it starts its cluster head selection timer according to the priority determined by algorithm type and it declares itself as cluster head with the timeout and sends this information by CJ packet to its neighbors. During that waiting period before sending its own CJ packet, if a node receives another CJ packet from one of its neighbors, than it stops its cluster head selection timer and joins the cluster indicated by the CJ packet and start data transfer rounds.
- Member node selection: When cluster head selection timer expires, the node declares itself as a cluster head and broadcasts a CJ packet to invite members to its cluster. A

CJ packet consists of a cycle id, cluster id and sender id, depth and other fields as explained above in section 3.2. Nodes that receive a CJ packet join the cluster, and if the depth limit is not reached, it modifies the received CJ packet by incrementing the depth level value and invites other nodes to the cluster by broadcasting modified CJ packet. Member nodes only change the sender id, and depth level values.

Each of the proposed algorithms employs a different cluster formation approach. The algorithms and their cluster selection methods are as follows;

Algorithm 1:

1. Sink node broadcasts an IC packet with a distinct cycle id.
2. Nodes who receive this IC packet declare themselves as cluster heads and form their own clusters, by broadcasting a CJ packet to its neighbors.
3. Nodes who receive CJ packets check the cycle id field,
   - 3.1. If a node has already joined a cluster, it discards the CJ packet.
   - 3.2. If a node is unassigned,
     - 3.2.1. It becomes a member of a cluster,
     - 3.2.2. If the depth level of the node is not greater than or equal to the depth limit specified in the CJ packet, it rebroadcasts the CJ packet by incrementing the depth level.
     - 3.2.3. If the depth level of the node is greater than or equal to the depth limit specified in the CJ packet and node has not received an IC packet with the same cycle id before, node creates an IC packet and broadcasts it to neighbors.

Algorithm 2:

1. Sink node broadcasts IC packet with a distinct cycle id.
2. An unassigned node receiving an IC packet for the first time starts cluster head selection timer and broadcasts the same IC packet to its neighbors immediately.
   - 2.1. If the node receives another IC packet with the same cycle id until the timer expires, it discards the IC packet.

3. When a node's timer expires, the node elects itself as cluster head, and broadcasts a CJ packet to its neighbors to form a cluster.
4. A node receiving a CJ packet,
    4.1. Discards the packet if a node has already joined a cluster.
    4.2. Otherwise, it stops its cluster head selection timer, if any, and then,
        4.2.1. It becomes a member of a cluster,
        4.2.2. If the depth level is not greater than or equal to the depth limit specified in the CJ packet, it rebroadcasts the CJ packet by incrementing the depth level.
        4.2.3. If the depth level of the node is greater than or equal to the depth limit specified in the CJ packet and node has not received an IC packet with the same cycle id before, node creates an IC packet and broadcasts it to neighbors.

Algorithm 3 and Algorithm 4:

1. Sink node broadcasts an IC packet with a distinct cycle id.
2. An unassigned node receiving an IC packet starts its cluster head selection timer and broadcasts same IC packet to its neighbors.
    2.1. If the node receives another IC packet with the same cycle id until the timer expires, increments its neighbor counter and reinitializes its cluster head selection timer according to the neighbor count.
3. When a nodes' timer expires, the node elects itself as cluster head, broadcasts a CJ packet to its neighbors to form a cluster.
4. A node receiving a CJ packet,
    4.1. Discards the packet if a node has already joined a cluster.
    4.2. Otherwise, it stops its cluster head selection timer, if any, and then,
        4.2.1. It becomes a member of a cluster,
        4.2.2. If the depth level is not greater than or equal to the depth limit specified in the CJ packet, it rebroadcasts the CJ packet by incrementing the depth level.
        4.2.3. If the depth level of the node is greater than or equal to the depth limit specified in the CJ packet and node has not received an IC packet with the same cycle id before, node creates an IC packet and broadcasts it to neighbors.

Algorithm 5:

1. Sink node broadcasts an IC packet with a distinct cycle id.
2. An unassigned node receiving an IC packet starts its cluster head selection timer according to its energy level and broadcasts the IC packet to its neighbors.
    2.1. If a node receives another IC packet with the same cycle id until timer expires, it checks the energy level value in the packet.
        2.1.1. If the node has a lower or equal energy level than the energy level specified in the packet it stops its cluster head election timer.
        2.1.2. If the node has a higher energy level than the energy level specified in the packet, it discards the IC packets.
3. When a nodes' timer expires, node elect itself as cluster head and broadcasts a CJ packet to invite its neighbors to the cluster.
4. A node receiving a CJ packet,
    4.1. Discards the packet if a node has already joined a cluster.
    4.2. Otherwise, it stops its cluster head selection timer, if any, and then,
        4.2.1. It becomes a member of a cluster,
        4.2.2. If the depth level is not greater than or equal to the depth limit specified in the CJ packet, it rebroadcasts the CJ packet by incrementing the depth level.
        4.2.3. If the depth level of the node is greater than or equal to the depth limit specified in the CJ packet and node has not received an IC packet with the same cycle id before, node creates an IC packet and broadcasts it to neighbors.

3.7.2 Data Transfer

After cluster formation, the data transfer period starts. The same data transfer method is employed to transfer data from members to cluster heads and then from cluster heads to sink in each round.

- Data transfer: in this period nodes transmit their data to cluster head nodes with data transfer packets, directly or via other members. Cluster heads collect data transfer

packets coming from cluster members, and send an aggregated data transfer packet to the sink periodically in each round.

Data transfer period for cluster heads;

In data transfer period, cluster heads (CH) periodically aggregate and send their readings and data collected from the members. Each C2S packet transmission corresponds to a round. Cluster heads sends their C2S packets after **T** seconds of forming its cluster (CF) at t_0, and then sends its C2S packets to the sink periodically in every **T** seconds for the specified number of rounds as seen in Figure 3.8.

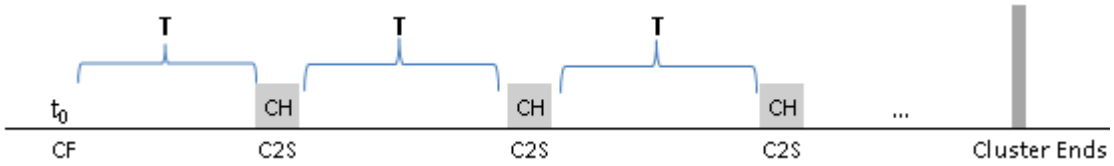


Figure 3.8: Cluster head with no members

Data transfer period for member nodes;

In data transfer period, members periodically aggregate and send their readings and the data sent by other member nodes. Each M2C packet transmission corresponds to a round. Members send their first M2C packet after T-($level * W$) seconds of joining a cluster at t_1, according to their level number. After the first M2C packet, they periodically send their M2C packets to their parents in every **T** seconds as seen in Figure 3.9.

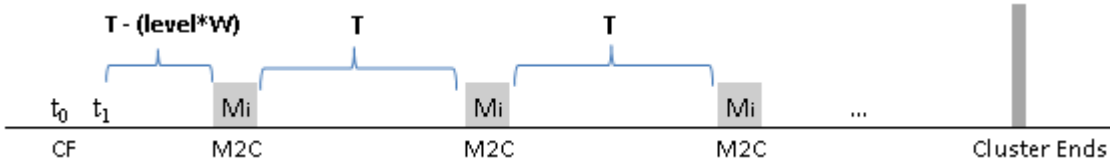


Figure 3.9: Cluster head with only one level member

For the above data transfer periods time parameters are transmitted to each node by IC and CJ packets and following equation must be always true to provide waiting time for proper data aggregation in nodes at different levels.

- $T > level * W$
- $level > 0$

At the end of each round M2C and C2S packets are transmitted to the cluster head and sink via the same path as the member and cluster head nodes receives the first IC and CJ packet, respectively. This allows us to collect data from the members and the cluster heads without requiring an additional routing algorithm. In Figure 3.10 we depicted an example flow of data transfer packets.

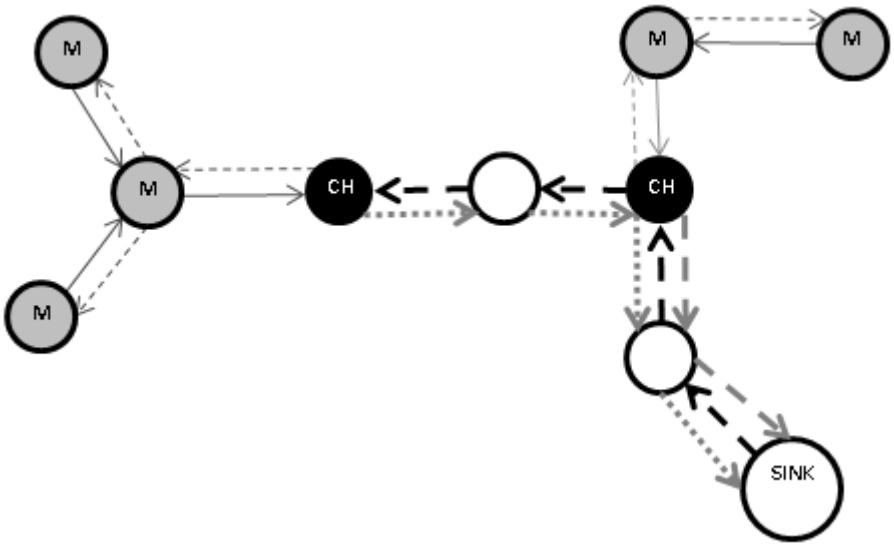


Figure 3.10: Message exchanges between members, cluster heads and sink

In this figure there are 2 different clusters, first cluster head is on the left and second cluster head is on the right. Straight lines show data transfer packets (M2C) from members to cluster heads in each round. Black dotted bold lines represent the path followed by CJ or IC packets. Gray dotted bold lines are the path of the first cluster head's C2S packet, and gray dashed bold lines are the path of the second cluster head's C2S packet to the sink at the end of each round.

In Chapter 4 implementation of experiments and the results of each experiment are discussed.

CHAPTER 4

IMPLEMENTATION RESULTS AND DISCUSSION

4.1 Experimental Setup

For the experimental setup, Crossbow IRIS motes [46] have been used[1]. A sample device is shown in Figure 4.1. Motes are connected to PC by using MIB520 USB interface board [47]. TinyOS-v2 [48] operating system is used to run algorithms in our experiments.

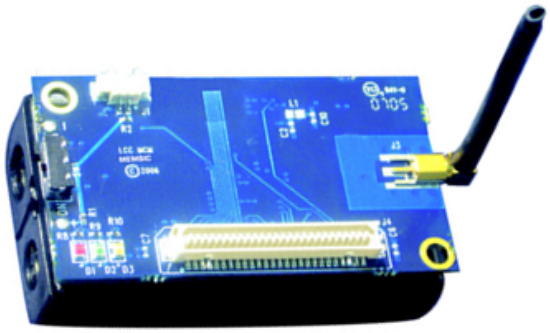


Figure 4.1: A real sensor node example

4.1.1 Hardware

Components of wireless sensor nodes are controller, communication device, memory and power supply. Each part has its own responsibilities to form an energy efficient, reliable sensor environment.

[1]The hardware [49] used in the experimental studies is supported by "Computer Networks and Wireless Technology Research Group" [50].

- *Controller* collects sensed data from sensors and processes it.
- *Communication device* used to exchange data between nodes with its wireless radio transceiver.
- *Memory* is used to store programs.
- *Power supply* is supplying power to components.

Some of the technical specifications of IRIS XM2110 motes used in the experiments are listed in Table 4.1.

Table 4.1: Features of IRIS XM2110 mote

Features	
Frequency Range	2.4 GHz ISM Band
Processor	Atmel ATMega 1281
Radio Transceiver	RF230 Atmel
Serial Flash	512 kB
RAM	8K bytes

4.1.2 Radio Transmission Power of IRIS motes

Radio transmission power is a programmable feature of IRIS motes. For IRIS motes transmission power is between 3dBm and -17.2dBm and each power level has a programmable code to use [51] which is shown in Table 4.2. For experiments we used power register codes 15 (PR15) and 13 (PR13). Highest power register code means lower coverage area.

Table 4.2: IRIS output power settings

RF Power (dBm)	milliWatts (mW)	Power Register (code)
3.0	1.995	0
0.5	1.122	5
-4.2	0.380	10
-9.2	0.120	13
-17.2	0.019	15

4.1.3 Software

In order to run algorithms we used TinyOS-v2 [48], which is an open source operating system designed for wireless sensors. Language used in TinyOS-v2 is called nesC [52], and it is an

extension of C language. TinyOS-v2, aims to use minimal resources of a node [7].

The following functions are implemented to realize our algorithms;

- **Timer fired function** In this function, nodes' duties when a timer expires are implemented. When timers are expired, functions stated for the timers starts, such as sending aggregated data packets or being a cluster head.
- **Packet received function** In this function, nodes' duties, when they received a packet, are implemented. Most of the computational processes are performed in this function. A different receive function is implemented for each algorithm.
- **Cluster formation function for cluster head nodes** Cluster creation for cluster heads differs in each algorithm. They employ different algorithms to form a cluster.
- **Data aggregation function for cluster heads and members** In case of data aggregation, cluster heads and members collect required data from its member nodes and calculate aggregation result to transmit it in a packet to the cluster heads and sink, respectively.

4.2 Topologies

We performed several experiments on three kinds of topologies.

4.2.1 Linear Topology - Hallway

Total of 23 nodes placed approximately 5 meters away from each other to cover 3 hallways of Informatics Institute's second floor as depicted in Figure 4.2. We divided and named nodes into three groups. Group 1 covers left hall and consists of nodes 2 - 9. Group 2 connects left and right halls and consists of nodes 10 - 17 and node 1. Group 3 covers right hall and consists of nodes 18 - 24. Sink node (node 1) and Listener node (node 0) is placed between node 13 and node 14.

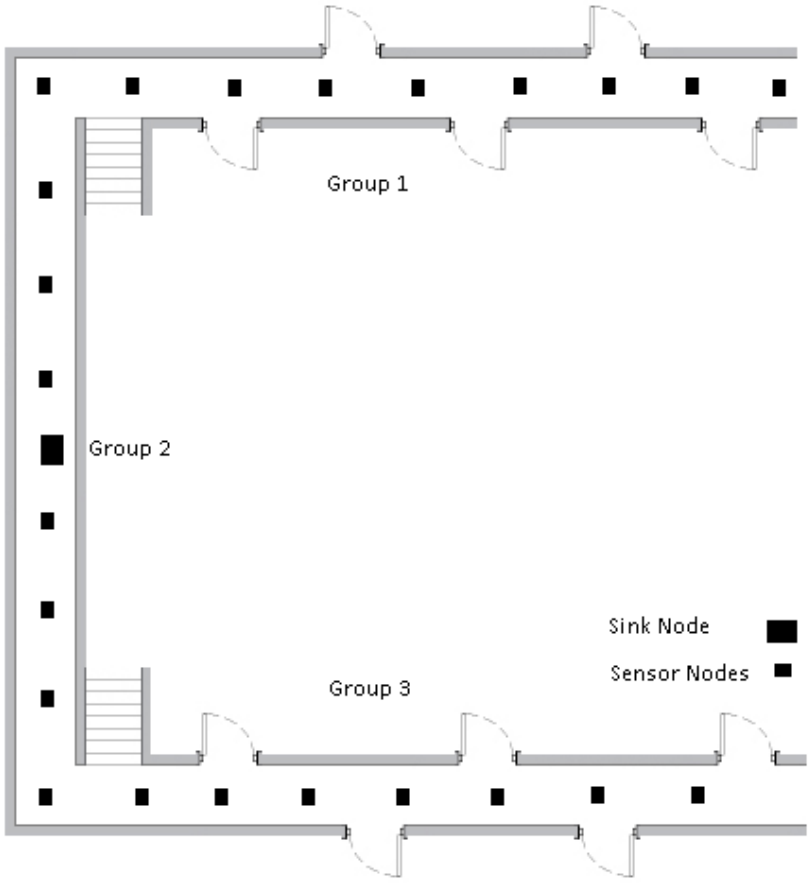


Figure 4.2: Hall topology node locations

In this figure big rectangle shows the location of sink node and small rectangles show location of sensor nodes.

4.2.2 Grid-1 Topology - Indoor

For this topology type we again used total of 23 nodes placed approximately 6 meters away from each other to cover Middle East Technical University Culture and Convention Center's foyer as depicted in Figure 4.3. We formed a grid area with 6 columns and 4 rows. Sink node (node 1) and Listener node (node 0) is placed between node 10 and node 14.

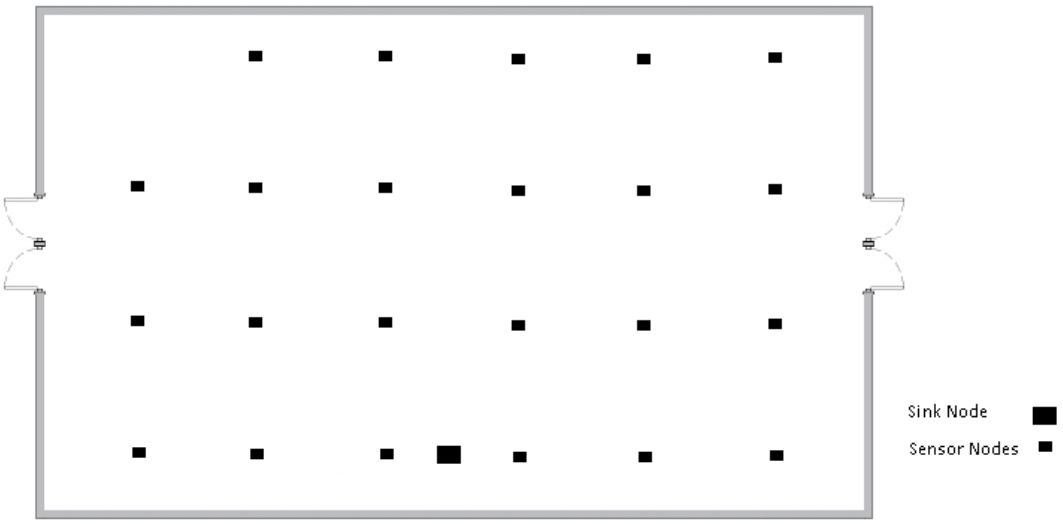


Figure 4.3: Grid-1 topology node locations

4.2.3 Grid-2 Topology - Outdoor

For this topology type we used total of 23 nodes placed approximately 6 meters away from each other to cover an orchard as depicted in Figure 4.4. Orchard is filled with trees and nodes placed between trees on the ground. We formed a grid area with 5 columns and 5 rows. Sink node (node 1) and Listener node (node 0) is placed in the middle of nodes 9, 10, 14 and 15.

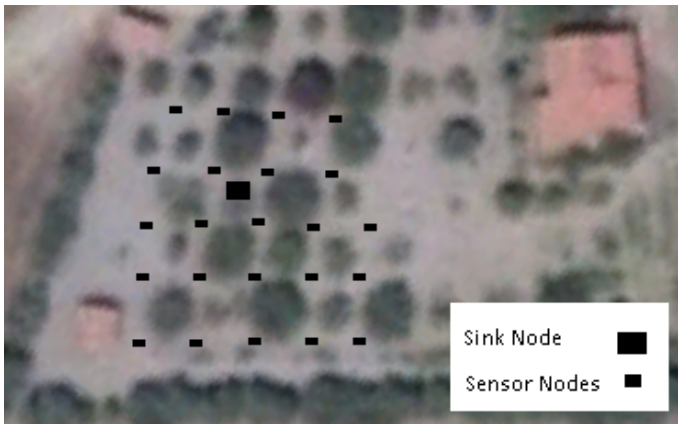


Figure 4.4: Grid-2 topology node locations

4.3 Data Collection

In order to collect exact statistics, we have introduced new a data collection period that takes place after the last cycle. The data collection period starts by a packet initiated by Sink node. However packet transmissions occur during this period not included in the statistics and they will not effect any results.

- Data collection period: in this period, nodes transmit their statistics of total transmitted and received packet counts, and their occurrence of being member or cluster head counts.

For the data collection the following packets are used;

- **Initiate-data-collection (IDC) packets:** An IDC packet is broadcasted at the beginning of data collection period and this packet inform nodes about the data collection period that is about to start. Sink node periodically re-sends IDC packet to reach all nodes. Following fields are included in IDC packets:
  - Initiator field - indicates that cycles are finished and data collection period will start,
  - Sender id - indicates id of the sender node.
  - Round count (N) - indicates round count of data collection period.
  - Wait time (W) - indicates how long (seconds) nodes should wait before sending next DC packet.
- **Data collection (DC) packets:** Data collection packets are used to collect desired data from all nodes at the end of each test via one hop - direct - communication. Nodes send their DC packets to the sink directly.
  - Sender id - indicates id of the sensor node.
  - Total transmitted packet count - indicates node's total transmitted packet count,
  - Total received packet count - indicates node's total received packet count,
  - Cluster head count - indicates number of occurrences of being a cluster head,
  - Member count - indicates number of occurrences of being a member.

Data collection period is same for all algorithms and nodes follow the following steps to send their data to sink;

1. Sink node broadcasts IDC packet with a distinct initiator.
2. When a nodes receives an IDC packet,
   2.1. If a node already received an IDC packet, discards the IDC packet.
   2.2. If it is the first time that a node receives and IDC packet, node starts its data collection period, and begins to send DC packets to Sink periodically.
3. DC packets received by Sink node transmitted to PC for evaluation.

In the following sections instead of using full algorithm names, we only used the following names for each algorithm.

- Algorithm 1 - Clustering based on first responders without waiting time,
- Algorithm 2 - Clustering based on first responders with random waiting time,
- Algorithm 3 - Clustering based on highest neighbor count,
- Algorithm 4 - Clustering based on lowest neighbor count,
- Algorithm 5 - Clustering based on energy level.

4.4 Parameters used in algorithms

We conducted the following experiments with the following parameters;

- We performed 3 tests for each algorithm. Each test consists of 3 cycles and each cycle consists of 15 rounds. Only in Grid-2 topology experiment we used 15 cycles and 3 rounds per test.
- The parameters we used in the algorithms are;
  - N=15 for the first 3 experiments, and N=3 for the last experiment,
  - T=4,
  - W=2,
  - D=2.

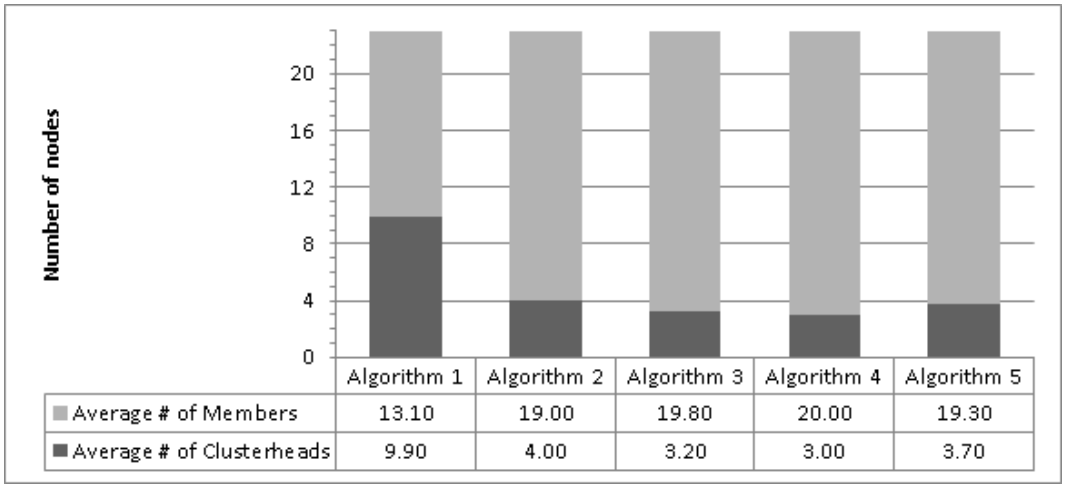
4.5 Results

In this section results obtained from the experiments on nodes are presented. Experiments have been done with 25 nodes; Listener node (Node 0), Sink node (Node 1) and 23 standard nodes (Node 2-24) have been used. Sink node is pre-assigned to Node 1 for all algorithms.

We performed four experiments on three different topologies for each of the five algorithms. We conduct one each experiment on two different transmission powers, PR13 and PR15, for linear topology and two experiments with PR15 for the two grid topologies. The experiments' results are presented and discussed below.

4.5.1 Experiments with PR13 using Linear topology

As the first experiment of our proposed algorithms we used PR13 transmission power and Linear topology. Each algorithm has been run for 3 times and the following results obtained. In the following, the averages of the results obtained from 3 tests are presented.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Average # of Members	13.10	19.00	19.80	20.00	19.30
Average # of Clusterheads	9.90	4.00	3.20	3.00	3.70

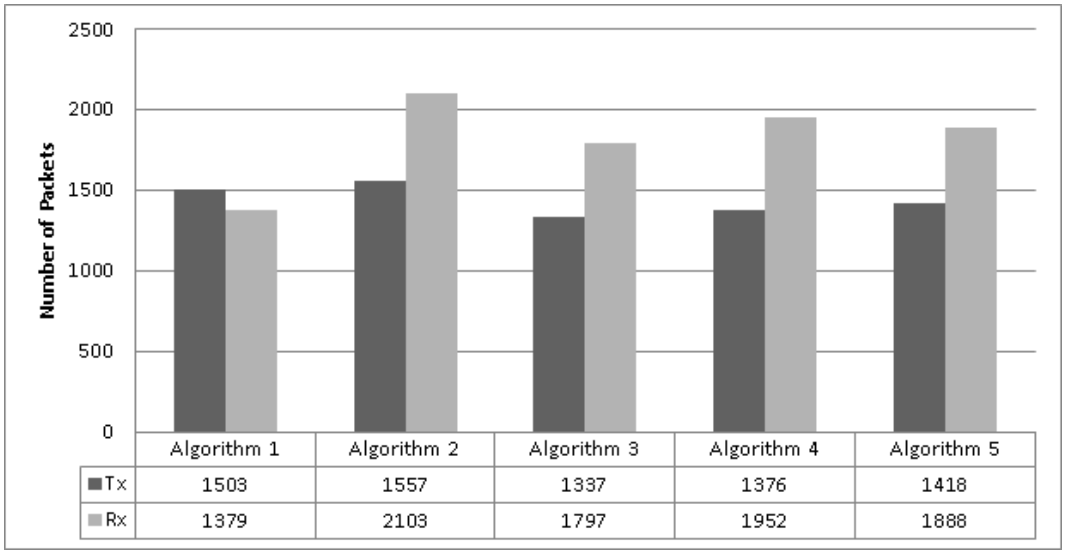
Figure 4.5: Average cluster head and member counts for PR13

For the average cluster head and member count graphs, average results represent average of 3 tests for each algorithm.

As seen from Figure 4.5, Algorithm 1 has the highest number of average cluster heads since all nodes who receive the IC packet declare itself as a cluster head immediately. With total of 23 nodes, Algorithm 1 has 9.90 average cluster head count.

In Algorithm 2, 3, 4 and 5, as nodes wait for a random time before declaring themselves as cluster heads and during that time they may join to other clusters, the number of cluster heads elected in the network is lower than Algorithm 1.

Results of Algorithm 2, 3, 4 and 5 very close to each other, and by looking just average cluster head counts we can say that Algorithm 4 is the best choice among them with 3.00 cluster head average per cycle for PR13 as it leads to minimum number of clusters in the network.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Tx	1503	1557	1337	1376	1418
Rx	1379	2103	1797	1952	1888

Figure 4.6: Total number of transmitted and received packets for PR13

Figure 4.6 shows the average total number of packets transmitted (Tx) and packets received (Rx) for different algorithms. Number of received packets and transmitted packet are proportional to member counts, depth of members, distance of cluster heads to sink and nodes transmit power strength.

As expected from results of cluster head and member counts, Algorithm 1 has higher average total transmitted packet count than its average total received packet count. This is because of the high number of cluster heads and less number of member nodes.

Results of Algorithm 3, 4 and 5 are very similar and Algorithm 3 has the lowest average total packet counts both in transmitted and received packets, 1337 and 1797, respectively. Having lower total packet counts means spending less energy in order to perform the same steps as other algorithms, which is forming clusters and collecting data from nodes.

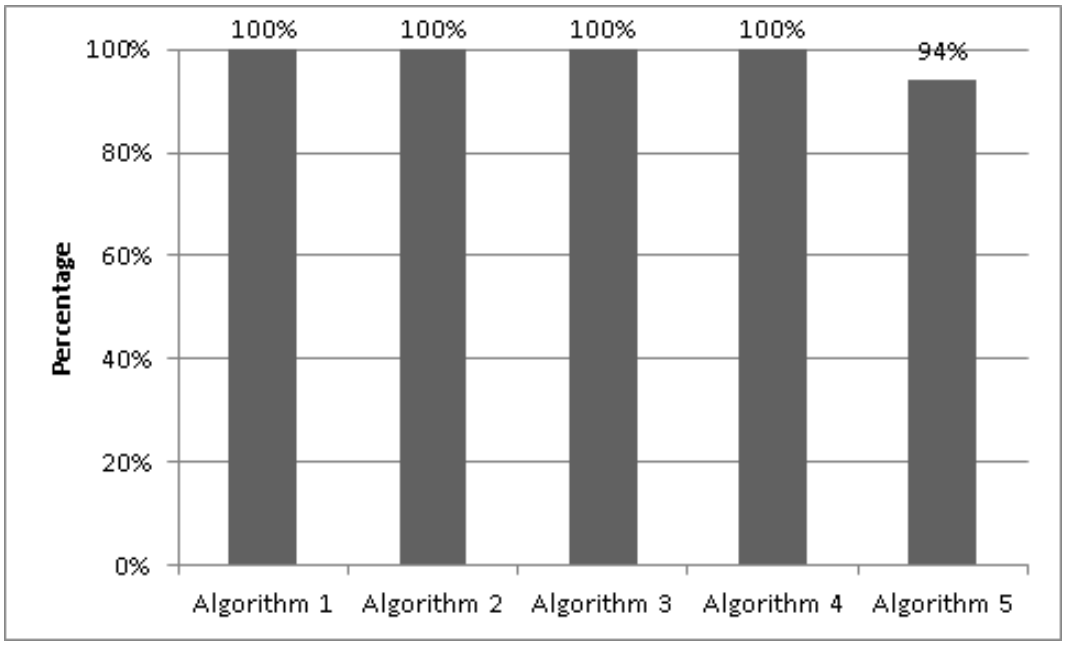


Figure 4.7: Coverage percentage of algorithms for PR13

Coverage graph shows us how much of the sensing area is covered by nodes. That is the ratio of nodes participate data collection by forming or joining a cluster. Results are average maximum coverage percentages of each test. When we perform our tests with PR13, four of the proposed algorithms have 100% coverage of the sensing area, and one of the algorithms has 94% coverage of the sensing area. This missing 6% is because in 2 of the tests, two nodes are unable to send their packets to any of the other nodes, although they become a member or a cluster head and even they had enough energy.

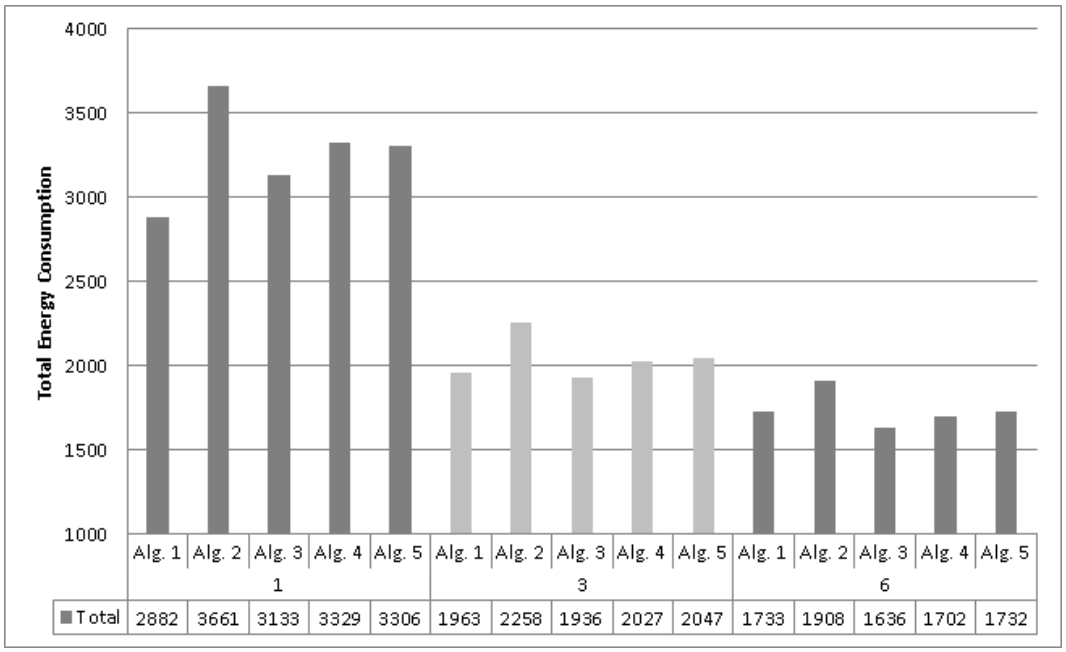


Figure 4.8: Energy consumption for different k values for PR13

This energy figure shows us total energy consumption based on transmitted (Tx) and received (Rx) packet counts, and k values determines how many Rx packets equals to Tx packets in terms of energy consumption. That is;

$$E_{TX} = k \cdot E_{RX}$$

In Figure 4.8, we used k values 1, 3 and 6. For the k values greater than 5, there is only a slight difference in the results. We calculated total energy consumption values with the following formula;

$$Total\ Energy\ Consumption = N_{TX} + \frac{N_{RX}}{k}$$

$k = 1$ to ∞

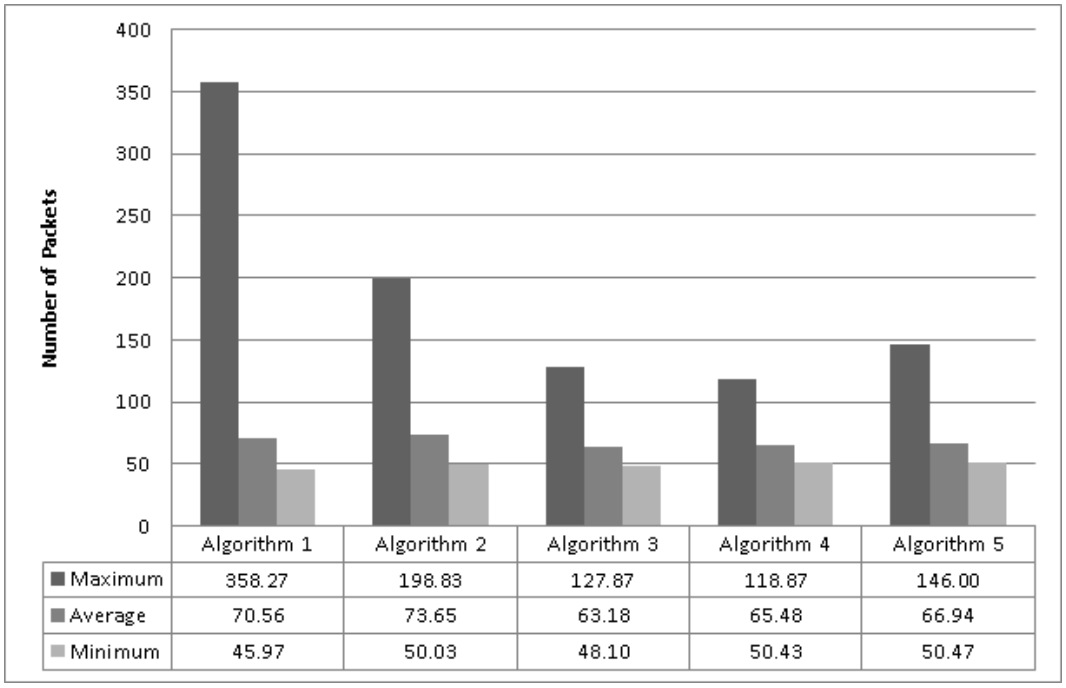
$T_X = Transmitted\ packet\ count$

$R_X = Received\ packet\ count$

As the results reveal, when k equals to 1, total energy consumption is less in Algorithm 1 as it has the lowest packet count. However when k is increased to 3, Algorithm 3 has the

lowest total energy consumption, because received packet count of Algorithm 3 is greater than Algorithm 1. Moreover when k is equals to 6 Algorithm 3, 4 and 5 has lower total energy consumptions than Algorithm 1. And finally Algorithm 2 has always the highest total energy consumption no matter what k value is, because it has the highest transmitted packet counts of all.

All in all, Algorithm 3 has lower energy consumption and Algorithm 2 has higher energy consumption than other algorithms.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Maximum	358.27	198.83	127.87	118.87	146.00
Average	70.56	73.65	63.18	65.48	66.94
Minimum	45.97	50.03	48.10	50.43	50.47

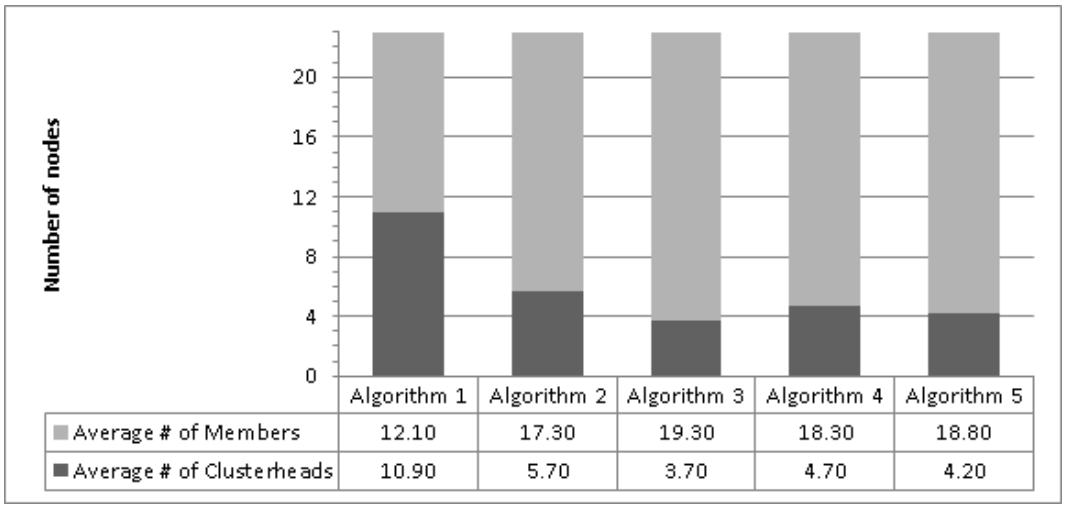
Figure 4.9: Maximum, Minimum and Average packet counts for $Tx + Rx/10$ equation for PR13

Figure 4.9 shows us, average results of maximum and minimum packet counts per node for each algorithm, according to $Tx + Rx/10$ equation. When the difference between average results of maximum and minimum packet count is lesser the results are better, because lower difference result means, network load is distributed evenly among nodes. For these results, Algorithm 4 has the lowest average difference with 68.40 packets.

However Algorithm 5 cannot ensure these results as expected, since its cluster head election procedure is based on the highest energy level according to the equation used in graph. Since our tests compose of only 3 cycles, Algorithm 5 has only 3 chances to minimize this difference, which is very low. Algorithm 5 may result better when tested for more cycles.

4.5.2 Experiments with PR15 using Linear topology

We have conducted the same set of tests in Linear topology but with different transmission power level.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Average # of Members	12.10	17.30	19.30	18.30	18.80
Average # of Clusterheads	10.90	5.70	3.70	4.70	4.20

Figure 4.10: Average cluster head and member counts for PR15

Figure 4.10 shows average cluster head and member counts per cycle for PR15. Again as explained in PR13, Algorithm 1 has the highest average cluster head count with 10.90 which is almost 2 to 3 times higher than other results.

Average cluster head counts are increased a little bit when compared to results for PR13. That increase is the result of transmission power decrease. Nodes are covering less area; therefore we need more cluster heads than PR13 to cover the sensing area.

For PR15, Algorithm 3 has the lowest average cluster head count among all, and by just looking for average cluster head and member counts we can say that Algorithm 3 is the best choice.

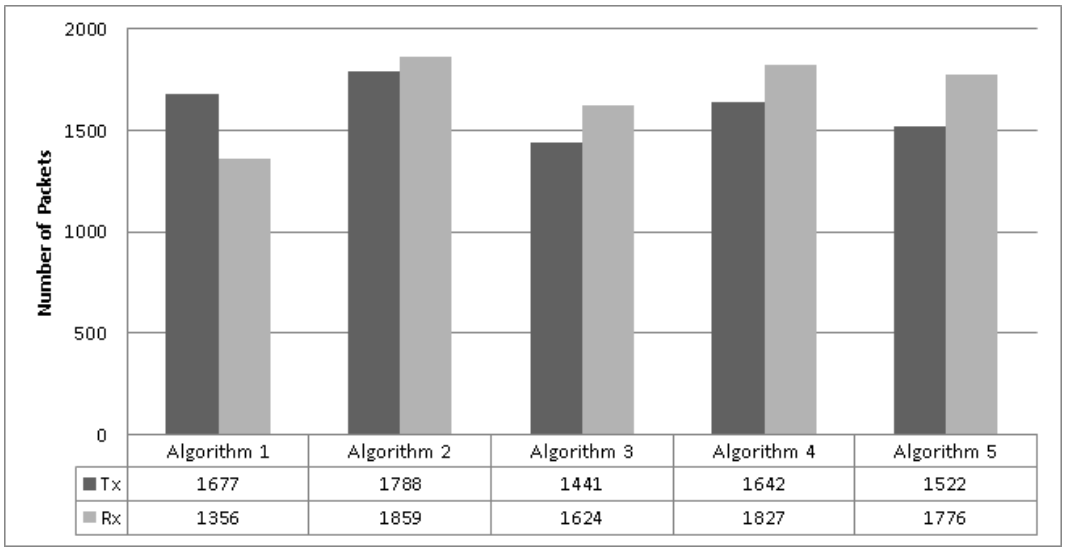


Figure 4.11: Total number of transmitted and received packets for PR15

As expected, Figure 4.11 shows that, Algorithm 1 has a higher average total transmitted packet count than its average total received packet count again. As for the remaining algorithms, Algorithm 3 has the lowest average total transmitted and average total received packet counts, 1441 and 1624, respectively.

Average total transmitted packet counts are increased and average total received packet counts are decreased when compared to the results for PR13. This is an expected result because when transmit power is decreased nodes send more packets to reach sink or cluster heads. Moreover since their transmit power is decreased, they are not able to hear as many packets as in PR13.

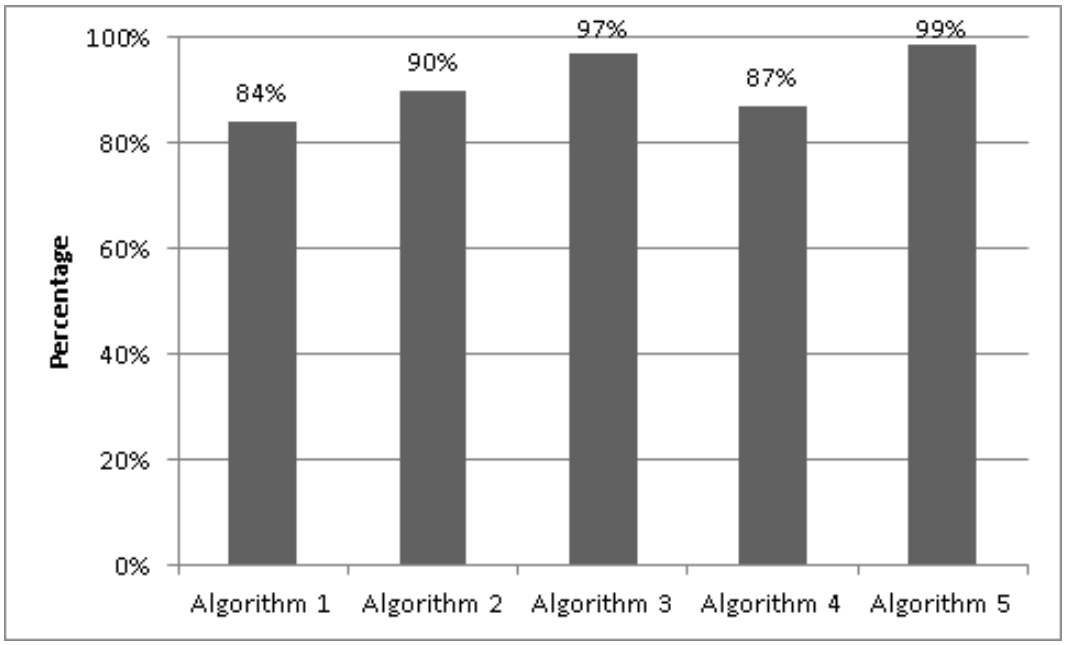


Figure 4.12: Coverage percentage of algorithms for PR15

Since nodes transmit power is decreased by 2 levels from PR13 to PR15, our coverage percentage gradually decreased. But this decrease does not means that nodes are not able communicate with other nodes. As we saw the average cluster head - member count graph above, all nodes are able to elect themselves as cluster heads or members in each cycle. However they experience some trouble in sending their packets or their packets got lost, dropped or collided with other packets. This is the main cause for this decrease in coverage results.

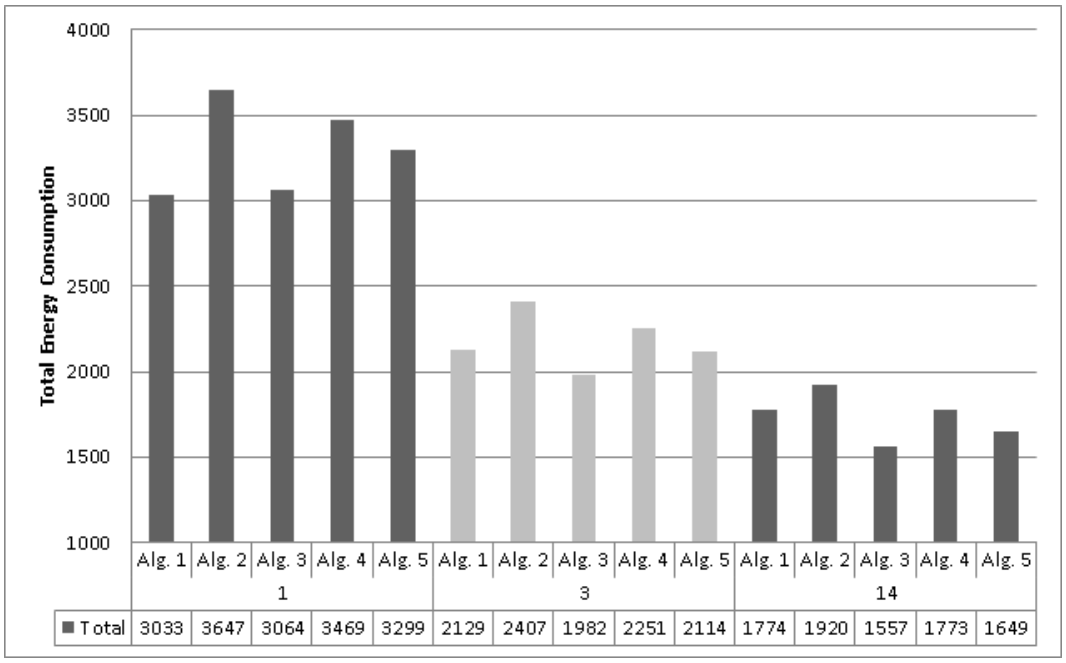
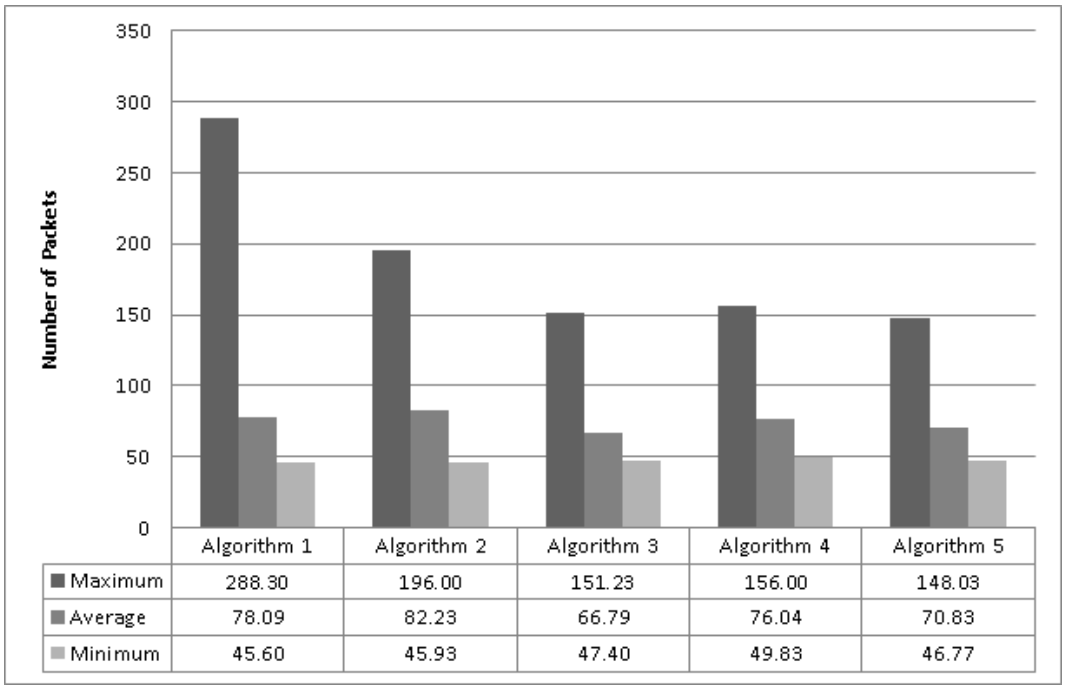


Figure 4.13: Energy consumption for different k values for PR15

Energy graph in Figure 4.13 shows that when k equals to 1 Algorithm 1 has the lowest total packet count among all algorithms. However when we increase k value, Algorithm 3 has the lowest total count. Moreover when k equals to 14, Algorithm 3, 4, and 5 has the lower total counts than Algorithm 1.

As Algorithm 2 has the highest transmitted packet count among all, again it has the highest total energy consumption regardless of k and Algorithm 3 is the best algorithm with $k \geq 3$.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Maximum	288.30	196.00	151.23	156.00	148.03
Average	78.09	82.23	66.79	76.04	70.83
Minimum	45.60	45.93	47.40	49.83	46.77

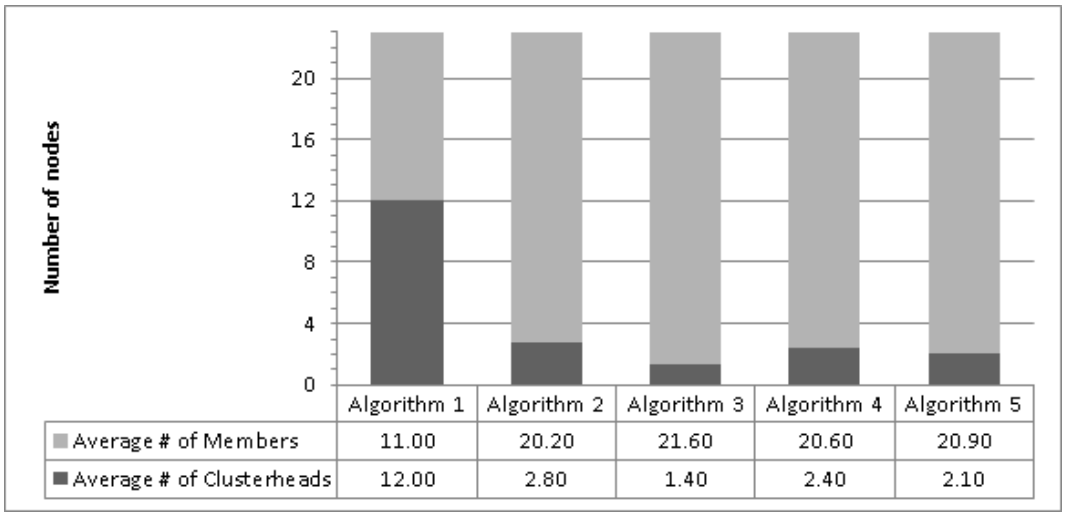
Figure 4.14: Maximum, Minimum and Average packet counts for $Tx + Rx/10$ equation for PR15

When we look at the difference between average results of maximum and minimum packet count values per node in Figure 4.14, Algorithm 5 has the lowest difference with 101.30 packets. Moreover Algorithm 5 has the lowest average results of maximum packet count among all, and this is also a good sign to show effectiveness of Algorithm 5. Also Algorithm 3 has nearly similar results as Algorithm 5.

We aimed to minimize average results of difference among all nodes in order to disseminate energy loads among nodes evenly. When we have less difference between maximum and minimum packet counts, we can say that our algorithms can disseminate load balance evenly among all nodes.

4.5.3 Experiments with PR15 using Grid-1 Topology

As the third set of experiments we have conducted the same set of tests again in a grid topology. We conducted our tests with power register code 15 and results of Grid-1 topology shown in Figure 4.15, 4.16, 4.17, 4.18 and 4.19.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Average # of Members	11.00	20.20	21.60	20.60	20.90
Average # of Clusterheads	12.00	2.80	1.40	2.40	2.10

Figure 4.15: Cluster head and member counts for PR15 Grid-1 Topology

Figure 4.15 clearly shows us that Algorithm 1 has the highest average cluster head count of 12.00, which is almost 9 times higher than the lowest average cluster head count.

Average cluster head counts are decreased when compared to hall topology type tests. Because in Grid-1 topology nodes are able to cover wider areas, therefore fewer nodes are sufficient enough to reach most of the nodes.

Results are almost similar to each other with slight differences, and Algorithm 3 has the lowest average cluster head count among all with 1.40 cluster heads per cycle.

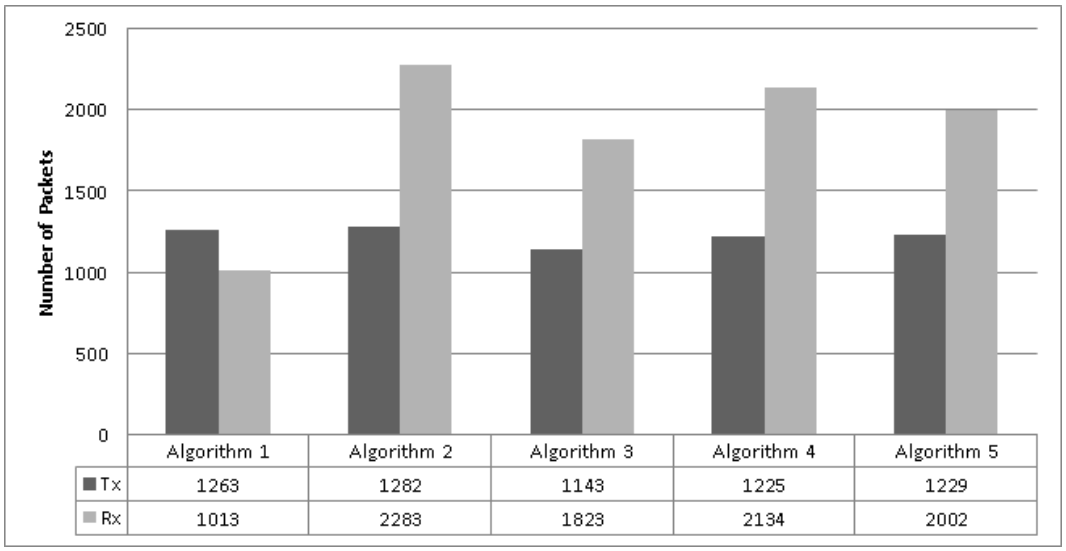


Figure 4.16: Total number of transmitted and received packets for PR15 Grid-1 Topology

Figure 4.16 shows that Algorithm 1 has higher average total transmitted packet count than its average total received packet counts. And again Algorithm 3 has the lowest average total transmitted and average total received packet counts of all, with 1143 and 1823, respectively.

As it is understood from Figure 4.16, Grid-1 topology affects the results significantly. Algorithms have the lowest average total transmitted packet counts and the highest average total received packet counts. Since nodes can cover wider areas, they can reach each other with fewer packets, therefore their average total transmitted packet counts are decreased significantly. Moreover since they can listen wider areas, their average total received packet counts are higher than other results.

As a result of these changes, difference between average total transmitted and average total received packet counts is higher than usual.

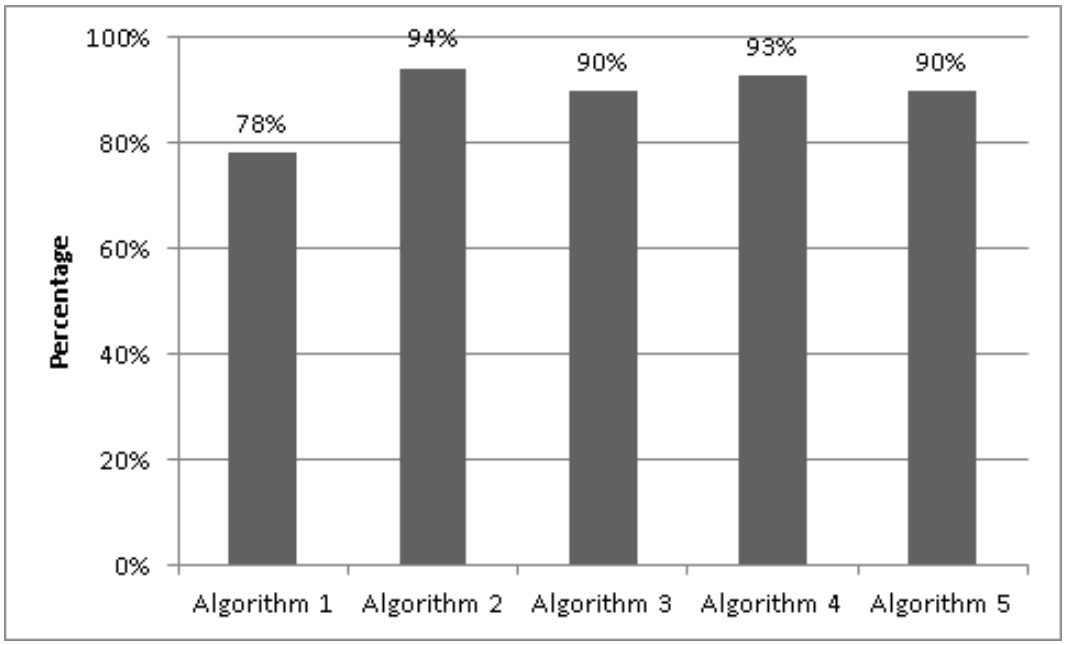


Figure 4.17: Coverage percentage of algorithms for PR15 Grid-1 Topology

Figure 4.17 shows the coverage results for all algorithms. As nodes can cover wider areas in Grid-1 topology, we expected that coverage results should be better because nodes can listen most of the network. However as their listening coverage is increased, probability of collisions are increased too. Therefore, despite nodes cover more than their sensing area, they are unable to cover it successfully.

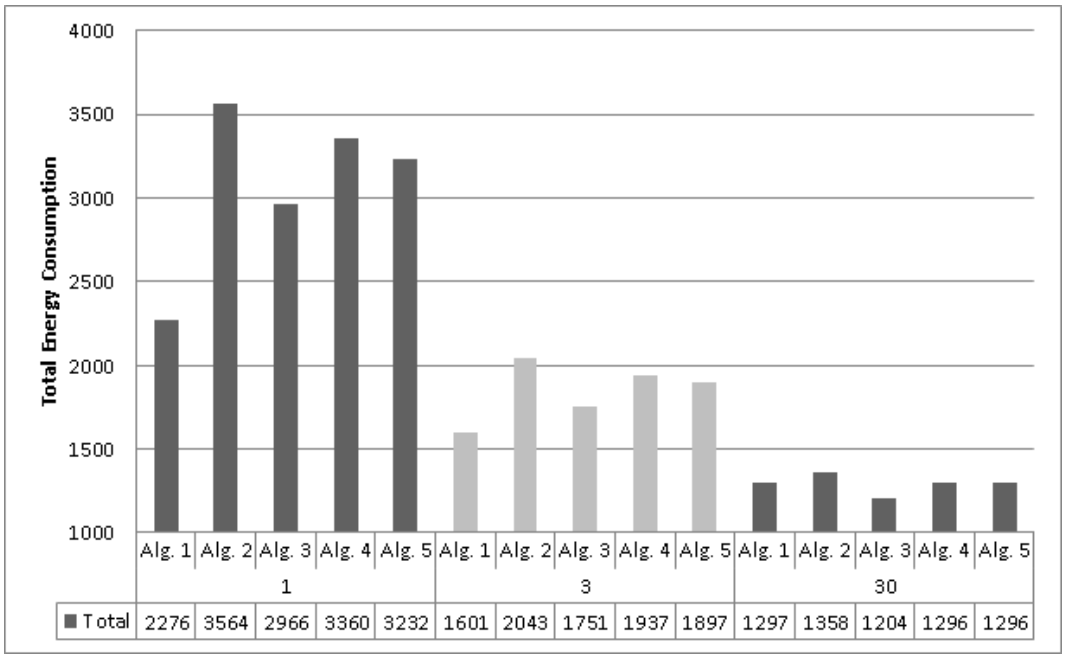
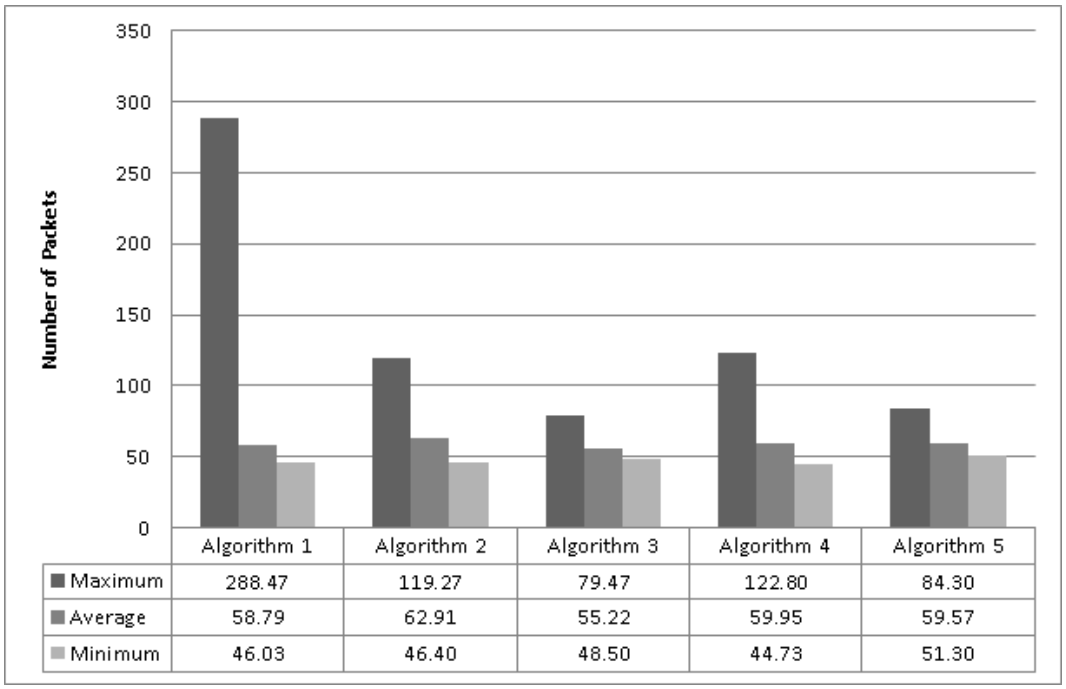


Figure 4.18: Energy consumption for different k values for PR15 Grid-1 Topology

Figure 4.18 shows the energy consumptions for all algorithms. When k equals to 1, energy graph again shows us Algorithm 1 has the lowest total count of all, and also difference between other results are significantly high. But when we change k value to 3, differences between other algorithms and Algorithm 1 decreases significantly.

When k is equals to 30, Algorithm 3, 4, and 5 has the lowest total counts than Algorithm 1. As for Algorithm 2, it has still has the highest total count of all, because it has the highest transmitted packet counts.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Maximum	288.47	119.27	79.47	122.80	84.30
Average	58.79	62.91	55.22	59.95	59.57
Minimum	46.03	46.40	48.50	44.73	51.30

Figure 4.19: Maximum, Minimum and Average packet counts for $Tx + Rx/10$ equation for PR15 Grid-1 Topology

Figure 4.19 shows that, the difference between average results of maximum and minimum packet count value is less in Algorithm 3, which is 31.0. Also Algorithm 5 has very similar results as Algorithm 3.

Since Algorithm 1 has no waiting time and no cluster head election procedure like other algorithms it has the highest difference between average results of maximum and minimum packet count values.

4.5.4 Experiments with PR15 using Grid-2 topology

As a final set of experiments we have conducted the same set of tests again with power register code 15, but this time we used Grid-2 topology.

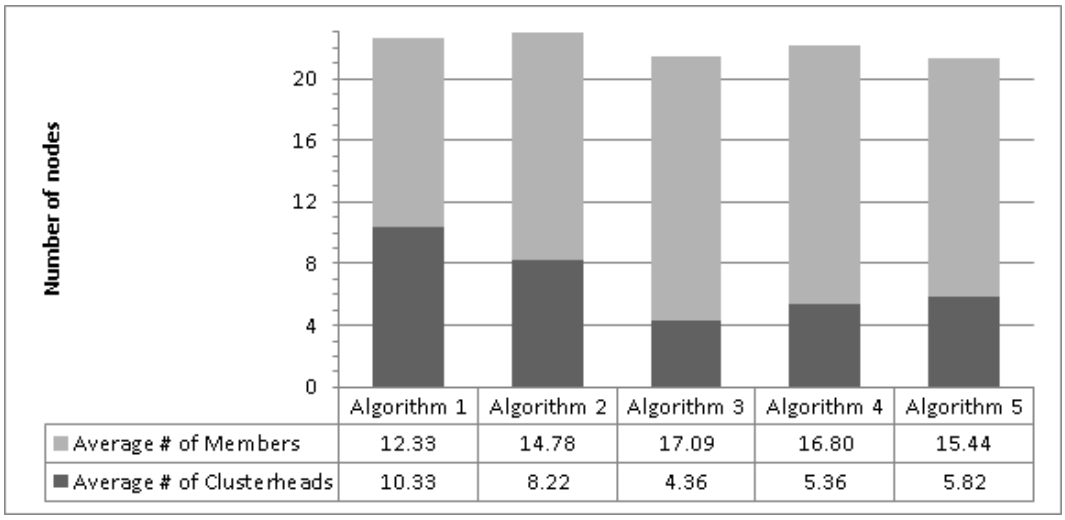


Figure 4.20: Cluster head and member counts for PR15 Grid-2 Topology

Figure 4.20 shows average cluster head and member counts for our algorithms after 3 tests. When compared to the results in Grid-1 topology since nodes' coverage is decreased in Grid-2 topology, average cluster head counts are increased for Algorithm 2, 3, 4 and 5, and cluster head count is decreased for Algorithm 1.

Moreover since there are trees among nodes during Grid-2 topology tests, some nodes are failed to receive cluster-join packets from their neighbors. As a result, not all 23 nodes could join a cluster all the time. Average total connected member counts are 22.7, 23.0, 21.4, 22.2, and 21.3, respectively for Algorithm 1, 2, 3, 4 and 5.

As a consequence, again, Algorithm 3 has the lowest average cluster head count among other algorithms with 4.36 cluster heads per cycle.

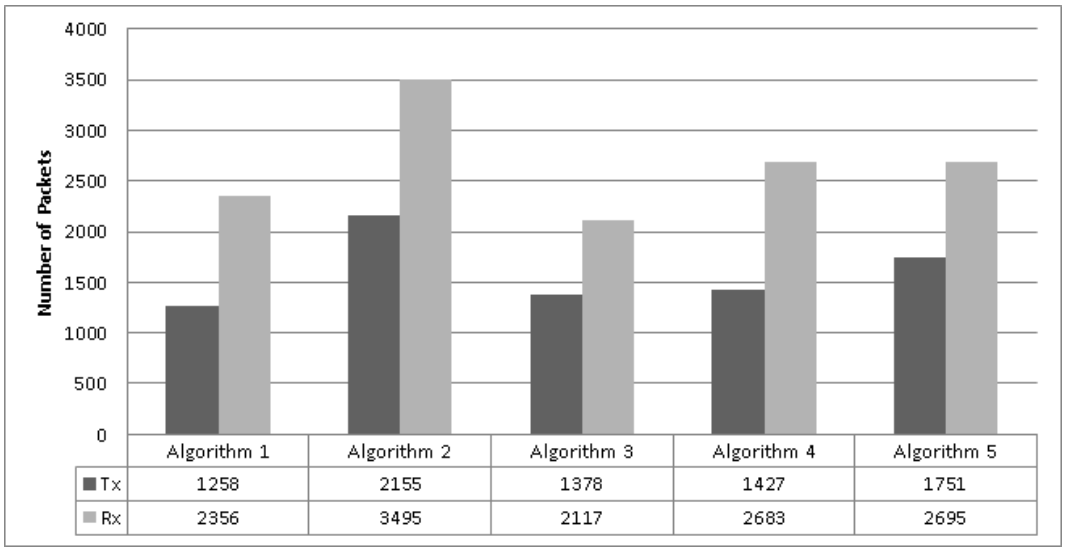


Figure 4.21: Total number of transmitted and received packets for PR15 Grid-2 Topology

Early tests resulted for Algorithm 1 as, higher average total transmitted packet counts than its average total received packet counts. Since we changed cycle and round numbers for Grid-2 topology tests, results of Algorithm 1 differs than its previous results. If we rule out Algorithm 1, Algorithm 3 has the lowest average total packet counts among other algorithms.

Results of Grid-2 topology tests in Figure 4.21 indicates the highest average total received packet counts among all four experiments. Since there is 15 cycles in one test, which is 5 times higher than previous tests, average total received packet counts are increased significantly.

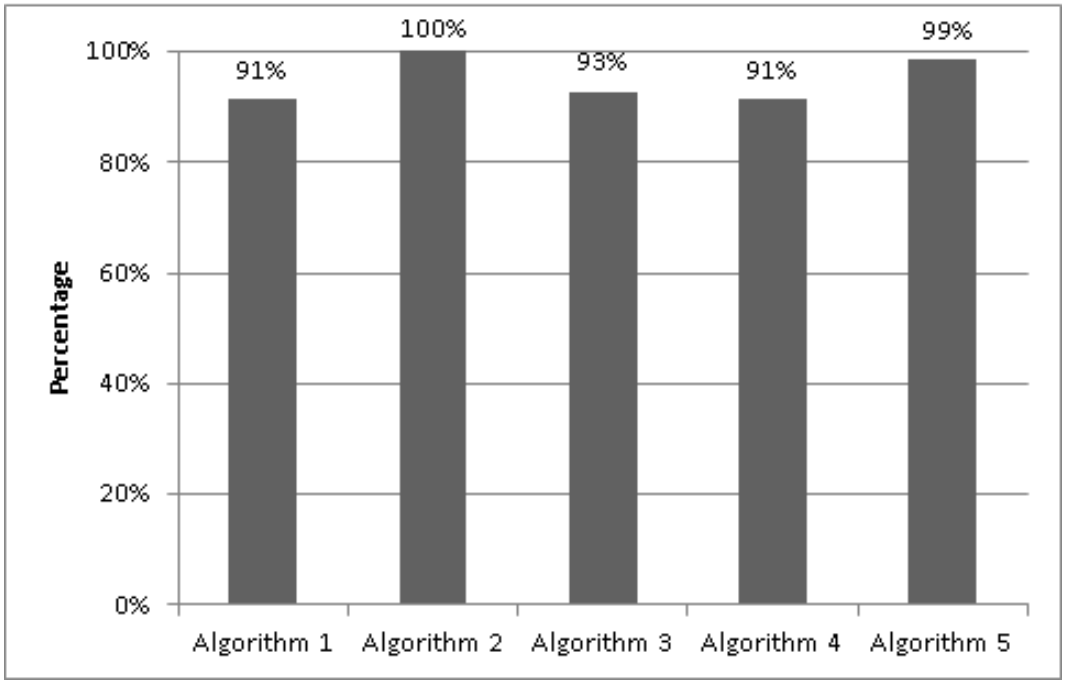


Figure 4.22: Coverage percentage of algorithms for PR15 Grid-2 Topology

Coverage results for Grid-2 topology shown in Figure 4.22 are very close for all algorithms. Since nodes coverage is not as larger as nodes coverage in Grid-1 topology, results are slightly better.

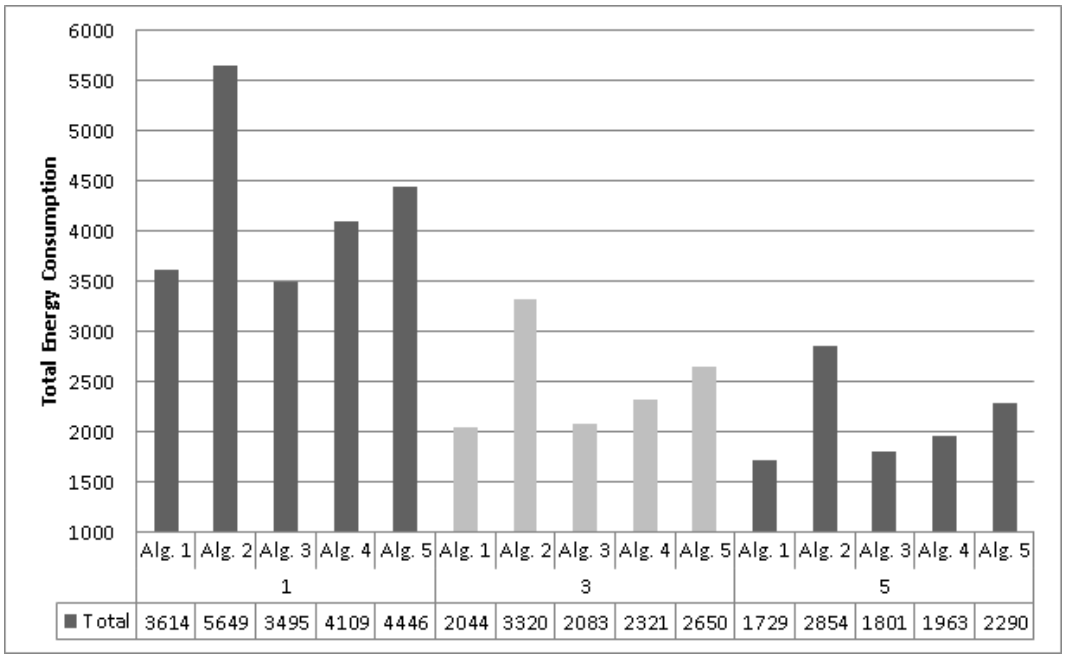
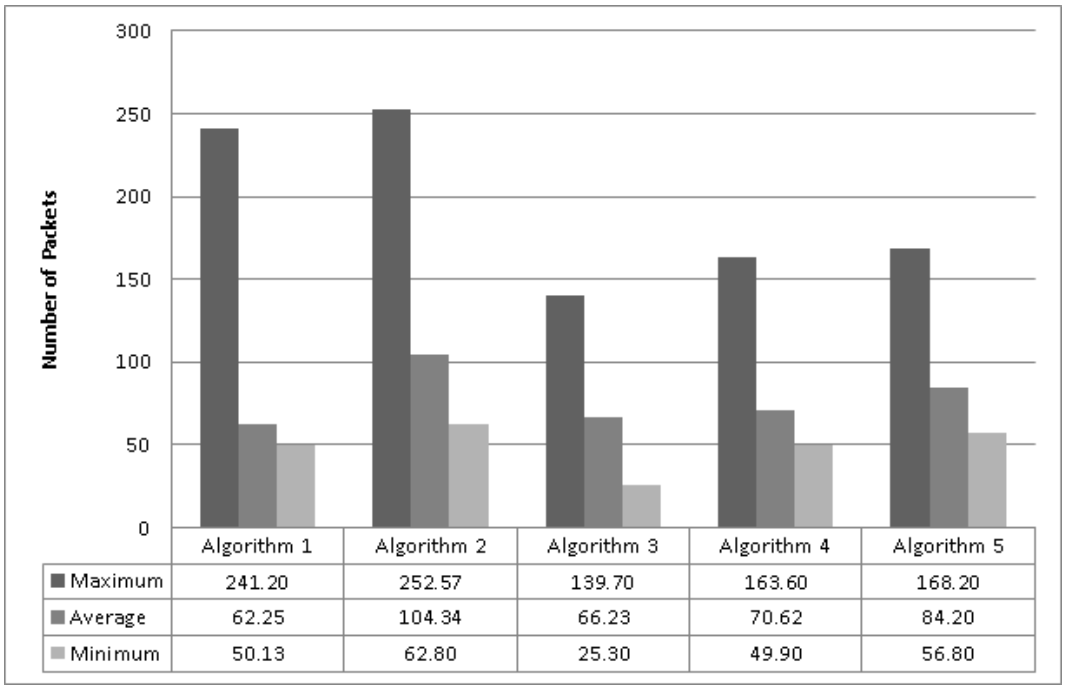


Figure 4.23: Energy consumption for different k values for PR15 Grid-2 Topology

Figure 4.23 shows that when k equals to 1, energy consumption graph shows us Algorithm 3 has the lowest total count of all. But when we change k value to 3 or greater, Algorithm 1 has the lowest energy consumption of all.

Algorithm 2 again has the highest energy consumption graph among all, because it has the highest transmitted and received packet counts.



	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Maximum	241.20	252.57	139.70	163.60	168.20
Average	62.25	104.34	66.23	70.62	84.20
Minimum	50.13	62.80	25.30	49.90	56.80

Figure 4.24: Maximum, Minimum and Average packet counts for $Tx + Rx/10$ equation for PR15 Grid-2 Topology

Figure 4.24 shows that, Algorithm 5 has the lowest difference between average results of maximum and minimum packet counts value which is 111.4. Also Algorithm 3 and 4 has very similar results as Algorithm 5.

For all the tests Algorithm 1 has always has the highest difference between its average maximum and minimum packet counts, which again reveal its poor performance among other algorithms. Since Algorithm 1 has no waiting time and no cluster head election procedure as other algorithms it is expected to has worse results than other algorithms we tested.

4.6 Discussions

When we compare all cluster head and member counts Algorithm 3 has the lowest cluster head counts of all. And for all of the experiments Algorithm 1 has the highest cluster head counts, which means nearly half of the nodes become cluster head.

Reason for this high cluster head count is, in Algorithm 1 nodes are designed to elect themselves as cluster heads after receiving IC packet immediately. Therefore when a node receives an IC packet, it immediately promotes itself as cluster head, and then propagates its CJ packet

to other nodes. However in other algorithms we inserted a short delay that allow smaller number of nodes to be cluster head and other nearby nodes to join that cluster. In Algorithm 2 nodes wait random time to promote themselves as cluster heads. In Algorithms 3, 4 and 5, nodes first rebroadcast a received IC packet to other nodes, and then each node decides their cluster head election priority according to the IC packets they received. Some nodes decided to cancel the cluster head election process and join the existing cluster and some nodes continue their cluster head election process to form their own clusters according to their priorities. At the end we have lower cluster head counts compared to that of Algorithm 1.

For the average total transmitted and average total received packet counts, Algorithm 1 has higher average total transmitted packet counts than its received packet counts, except in last experiment. This means Algorithm 1 consumes more energy to fulfill the same task compared to other algorithms. In most of the experiments Algorithm 3 has the lowest packet counts of all for both average total transmitted and average total received packets.

When we look at the coverage results, we can see that almost all algorithms have 100% coverage of the network at least once in the experiments. Lower results are mainly caused by lost, dropped or collided packets.

For the energy consumption results, again Algorithm 3 almost has the lowest counts of all. Energy graph is directly proportional to transmitted packet counts. Since Algorithm 2 has always had the highest average total transmitted packet counts, it also has the highest energy consumption of all.

And for the average result of maximum-minimum total packet counts, we expected Algorithm 5 have the minimum differences among all. However since our algorithms only run for 3 cycles, Algorithm 5 cannot perform well always compared to other algorithms. Therefore we conduct last experiment with 15 cycles and 3 rounds, to test effectiveness of Algorithm 5. Results indicates that Algorithm 5 has the lowest difference between its maximum and minimum total counts, but they are not very significant when compared to other algorithms. In order to see significant differences among results of Algorithm 5 and other algorithms, we need to increase number of cycles of tests.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis we have conducted several experiments to evaluate the performances of clustering approaches for sensor networks. We proposed five different clustering algorithms with different cluster head selection algorithms. We considered three types of sensor nodes: Sink node, cluster heads and member nodes. Role of the Sink node is pre-assigned and not changed throughout the network operation. However other nodes determine their roles on run-time, whether they become cluster heads or member nodes according the clustering algorithm employed.

Most of the previous studies in this field were performed using simulation tools. However, in this thesis, we performed our tests with real sensor hardware. Since our tests took longer time to complete than tests performed by simulations, we can only perform limited number of tests. Hence we performed experiments on 3 different topologies and two different transmission ranges.

Tests on different sensing areas produce very different results. In some places, nodes have larger transmission range, and in other places nodes have smaller transmission range. We performed some experiments on narrow corridors with smaller transmission ranges. And we also performed experiments on a large field with larger transmission range of nodes.

As results indicate Algorithm 1 and 2 have higher energy consumption than other algorithms. In Algorithm 1 too many nodes become cluster heads, which lowers the number of members per cluster. Moreover Algorithm 1 has higher transmitted packet count than its received packet count. Results of Algorithm 2, is slightly better than Algorithm 1 when compared the cluster head counts. However Algorithm 2 has the highest received packet count among all tests, which increases the energy consumption of nodes.

When we compare results of Algorithm 3, 4 and 5 we can easily figure out that their results are very similar to each other with very little differences. However Algorithm 3 has the best results among all. Between these three algorithms selecting the best one is very hard decision to make, because each algorithm has its own advantages and selecting the best logical one is possible when we know the desired properties of the application. For example if we want an algorithm to minimize energy differences among nodes Algorithm 5 is the best choice. Moreover if we want large clusters close to borders of the coverage area, Algorithm 4 is the best possible choice. Lastly, Algorithm 3 chooses large clusters close to the center of the coverage area, where nodes have too many neighbors.

As we stated before, with high network coverage, very low network traffic and high number of nodes per cluster Algorithm 3 is the best choice among our proposed algorithms. Before performing tests, our prediction of best energy efficient algorithm among all is either Algorithm 3, 4 or 5. Because these algorithms implemented cluster head selection feature based on some parameters, such as neighbor count or energy level.

5.1 Future Work

We defined several algorithms with different features, and as a future work, we can combine some of the features of these algorithms to select cluster heads more efficiently. For example, we can select cluster heads from nodes with more neighbors and higher energy levels. As a result, we might have both accomplish large cluster sizes and minimal energy level differences among nodes.

Moreover we can perform our algorithms on more challenging topologies with higher transmission ranges. Therefore we can find out effects of obstacles among nodes, such as trees, walls, or elevation differences etc.

Also we can perform same tests on simulation tools by incrementing number of tests. Therefore we can compare results with simulation and our experiments. As a result we can support our findings with the results of simulation tools.

We implemented pre-coded queuing system that comes with TinyOS-v2, but nodes still dropped some of the packets, because of high density of packet flow. Also we might consider imple-

menting a transmission schedules for each node for packet collisions. Therefore we can be sure packet deliveries by minimizing packet collisions.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, March 2002.

[2] K. Romer and F. Mattern. The design space of wireless sensor networks. *Wireless Communications, IEEE*, 11(6):54 – 61, December 2004.

[3] K. A. Delin, S. P. Jackson, D. W. Johnson, S. C. Burleigh, R. R. Woodrow, J. M. Mcauley, J. M. Dohm, F. Ip, T. P. A. Ferré, D. F. Rucker, and Others. Environmental studies with the sensor web: Principles and practice. *Sensors*, 5(1-2):103–117, February 2005.

[4] B. Son, Y. Her Her, and J. Kim. A design and implementation of forest-fires surveillance system based on wireless sensor networks for south korea mountains. volume 6, pages 124–130, September 2006.

[5] A.S. Tanenbaum, C. Gamage, and B. Crispo. Taking sensor networks from the lab to the jungle. *Computer*, 39(8):98 –100, August 2006.

[6] Upkar Varshney. Pervasive healthcare and wireless health monitoring. *Mob. Netw. Appl.*, 12(2-3):113–127, March 2007.

[7] Aleksandar Milenković, Chris Otto, and Emil Jovanov. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Comput. Commun.*, 29(13-14):2521–2533, August 2006.

[8] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. *Pervasive Computing, IEEE*, 3(1):38 – 45, January 2004.

[9] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. pages 108 – 120, January 2005.

[10] Lan Wang and Yang Xiao. A survey of energy-efficient scheduling mechanisms in sensor networks. *Mob. Netw. Appl.*, 11(5):723–740, October 2006.

[11] O. Younis, M. Krunz, and S. Ramasubramanian. Node clustering in wireless sensor networks recent developments and deployment challenges. *Network, IEEE*, 20(3):20 – 25, May-June 2006.

[12] Cuneyt Sevgi. *Network Dimensioning in Randomly Deployed Wireless Sensor*. PhD thesis, Middle East Technical University, 2009.

[13] O. Younis, M. Krunz, and S. Ramasubramanian. Node clustering in wireless sensor networks: recent developments and deployment challenges. *Network, IEEE*, 20(3):20 – 25, May-June 2006.

[14] Haowen Chan and Adrian Perrig. Ace: An emergent algorithm for highly uniform cluster formation. In *in Proceedings of the First European Workshop on Sensor Networks (EWSN*, pages 154–171, 2004.

[15] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks a survey. *Wireless Communications, IEEE*, 14(2):70 –87, April 2007.

[16] A. Boukerche, R.W.N. Pazzi, and R.B. Araujo. Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol. pages 560 –567, November 2005.

[17] Olivier Powell, Pierre Leone, and José Rolim. Energy optimal data propagation in wireless sensor networks. *J. Parallel Distrib. Comput.*, 67(3):302–317, March 2007.

[18] H. Dilum Bandara and A.P. Jayasumana. An enhanced top-down cluster and cluster tree formation algorithm for wireless sensor networks. pages 565 –570, August 2007.

[19] Seema Bandyopadhyay and E.J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. volume 3, pages 1713 – 1723 vol.3, March-April 2003.

[20] D. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *Communications, IEEE Transactions on*, 29(11):1694 – 1701, November 1981.

[21] Wendi B. Heinzelman, Anantha P. Ch, and Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1:660–670, Otober 2002.

[22] Wendi Beth Heinzelman. *An Application-Specific Protocol Architecture for wireless Networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

[23] Timothy J. Shepard. A channel access scheme for large dense packet radio networks. *SIGCOMM Comput. Commun. Rev.*, 26(4):219–230, October 1996.

[24] Stephanie Lindsey, Cauligi Raghavendra, and Krishna M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):924–935, September 2002.

[25] M.J. Handy, M. Haase, and D. Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pages 368 – 372, 2002.

[26] S. Basagni. Distributed clustering for ad hoc networks. In *Fourth International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 310 –315, 1999.

[27] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 1028 –1037 vol.2, 2001.

[28] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 4 vol. (xxxv+2866), March 2004.

[29] Ad Hoc Networks, Mainak Chatterjee, Sajal K Das, and Damla Turgut. WCA : A Weighted Clustering Algorithm for Mobile. pages 193–204, April 2002.

[30] C.R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *Selected Areas in Communications, IEEE Journal on*, 15(7):1265 –1275, September 1997.

[31] N. Shacham, E. Craighill, and A. Poggio. Speech transport in packet-radio networks with mobile nodes. *Selected Areas in Communications, IEEE Journal on*, 1(6):1084 – 1097, December 1983.

[32] C.R. Lin and M. Gerla. Asynchronous multimedia multihop wireless networks. volume 1, pages 118 –125, April 1997.

[33] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, mobile, multimedia radio network. *Wirel. Netw.*, 1(3):255–265, August 1995.

[34] G. Gupta and M. Younis. Load balanced clustering of wireless sensor networks. volume 3, pages 1848 – 1852, May 2003.

[35] Zhen Liu and Chen Zhigang. Data gathering in wireless sensor networks. In *Sixth International Conference on Semantics Knowledge and Grid*, pages 351 –354, November 2010.

[36] Choon-Sung Nam, Hee-Jin Jeong, and Dong-Ryeol Shin. The adaptive cluster head selection in wireless sensor networks. In *Semantic Computing and Applications, 2008. IWSCA '08. IEEE International Workshop on*, pages 147 –149, July 2008.

[37] F. Bouhafs, M. Merabti, and H. Mokhtar. A semantic clustering routing protocol for wireless sensor networks. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 351 – 355, January 2006.

[38] Liansheng Tan, Yumei Ying, and Wei Liu. Cote: A clustering scheme with optimal tiers and energy efficiency in wireless sensor networks. In *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, pages 164 –171, November 2010.

[39] A. Boukerche and A. Martirosyan. An energy-aware and fault tolerant inter-cluster communication based protocol for wireless sensor networks. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 1164 –1168, November 2007.

[40] Zhanyang Zhang and Guping Zheng. A cluster based query protocol for wireless sensor networks. volume 1, pages 140 –145, February 2006.

[41] The network simulator ns-2, August 2012. `http://www.isi.edu/nsnam/ns/`.

[42] P. Popovski, F.H.P. Fitzek, H. Yomo, T.K. Madsen, R. Prasad, and N.J. Vej. Mac-layer approach for cluster-based aggregation in sensor networks. pages 89 – 93, May-June 2004.

[43] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *Networking, IEEE/ACM Transactions on*, 11(1):2 – 16, February 2003.

[44] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. MSWiM '04, pages 157–164, New York, NY, USA, October 2004. ACM.

[45] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pages 90 –100, February 1999.

[46] Iris wireless measurement system data sheet, August 2012. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135%3Airis`.

[47] Mib 520 usb interface board datasheet, August 2012. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=144%3Amib520cb`.

[48] Vlado Handziski Philip Levis, David Gay et al. T2: A second generation os for embedded sensor networks. TKN Technical Report Series TKN-05-007, Telecommunication Networks Group, Technische Universität Berlin, November 2005.

[49] Classroom kit data sheet, August 2012. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=171%3Aclassroom-kits`.

[50] Computer Networks and Wireless Technology Research Group, August 2012. `http://www.ii.metu.edu.tr/tr/research_group/computer-networks-and-wireless-technology-research-group`.

[51] Mote processor radio and mote interface boards user manual, August 2012. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/6-user-manuals.html`.

[52] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesc language: A holistic approach to networked embedded systems. *SIGPLAN Not.*, 38(5):1–11, May 2003.