國立交通大學

資訊科學與工程研究所

碩 士 論 文

無線感測網路中利用空間關聯性
之節能資料收集方法

An Energy-Efficient Data Gathering Scheme Exploiting
Spatial Correlation in Wireless Sensor Networks

研 究 生：陳冠翰

指導教授：黃俊龍 教授

中 華 民 國 九 十 六 年 九 月

無線感測網路中利用空間關聯性之節能資料收集方法
An Energy-Efficient Data Gathering Scheme Exploiting Spatial Correlation in Wireless Sensor Networks

研 究 生：陳冠翰　　Student：Kuan-Han Chen

指導教授：黃俊龍　　Advisor：Jiun-Long Huang

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年九月

無線感測網路中利用空間關聯性之節能資料收集方法

學生：陳冠翰　　指導教授：黃俊龍

國立交通大學 資訊科學與工程研究所

摘　　要

隨著科技的發展與網路的普及，無線感測網路日益展露其重要性與實用性。因此，本論文在無線感測網路中，提出一利用空間關聯性之節能資料收集方法。第一，我們利用空間關聯性將一大群感測節點分類為一個個叢集，每個叢集中的節點感測相似的環境讀數並僅挑選出一個代表點回報該叢集的讀數。第二，為了進一步節省能源的消耗，各叢集回報少量的資訊給伺服器，由伺服器重新選擇可合併的叢集，並把選擇組合叢集的方式模組化為 infer-graph set 問題，提出一貪婪式演算法去解決。第三，我們提供每個感測節點一個 safe region，避免經常性的溫度改變導致叢集的重建以增加叢集存在的持久性。實驗顯示本論文提出之方法明顯降低了資料傳輸量以及叢集的數目，因此增加了無線感測節點的生命週期，實驗結果並証明了提出之方法較適用於動態的感測環境。

An Energy-Efficient Data Gathering Scheme Exploiting Spatial Correlation in Wireless Sensor Networks

Student：Kuan-Han Chen　　Advisors：Jiun-Long Huang

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

In this thesis, we propose an energy-efficient data gathering scheme over a highly spatial-correlated region in wireless sensor networks. First, we present a mechanism to group sensor nodes into a set of clusters, which have monitored similar phenomena by exploiting spatial correlation. The member readings in a cluster are bounded within an user-tolerable threshold and each cluster is represented by one clusterhead to answer queries. Second, in order to further reduce energy consumption, we merge the clusters which got resembled readings. The problem to merge clusters is modeled as an infer-graph set problem. We devise a greedy-based heuristic algorithm to acquire the near-optimal solution of choosing the new representative clusters. Third, we design a safe region for each sensor (even clusterhead or member sensor) not to casually leave the cluster and strengthen the persistence of clusters. It avoids the overheads of frequently rebuilding clusters. Experimental results show that our work is superior to previous techniques by saving 11 percent of bytes count and decreasing 15 percent of clusterhead count. Furthermore, our scheme is more suitable for dynamic sensing environment than competing techniques.

誌　　謝

回想起過去兩年碩士班點點滴滴的學習歷程，如今得以順利的完成論文，首先誠摯的感謝指導教授 黃俊龍博士於課業及生活上的指導及幫忙，老師開明的指導風格，引導學生構思實驗的可能，使我在這些年中獲益匪淺，老師對學問的嚴謹更是我學習的典範。感謝畢業口試委員，彭文志博士以及劉傳銘博士撥冗蒞臨指導，提供寶貴的建議與指導，使本研究論文能更臻完善。

感謝工程三館 219 實驗室的成員們：博士班士銓學長；同學承恩、信翰及瑞男，在兩年的日子裡培養共同修課的革命情感以及學術上的討論洞悉，都讓我在觀察事物時有更精闢銳利的眼光；學弟妹壬禾、國禾、建平、欣怡、亭瑩，你們的陪伴與幫忙都讓我銘記在心。

本論文的完成最要感謝家父，您是我一生的榜樣，每當我軟弱無助向您求助時，您總是能給予我力量與希望，分析問題的困境並提出解決之道，讓我有勇氣處理問題不致迷失方向，有你這樣的父親是我永遠的驕傲。感謝家母在我論文遇到瓶頸時也常打電話給我加油鼓勵，為我向主禱告，讓我感受到主的溫暖與實際。兄長在工作之餘也會抽空關心我的近況，使我能在家人的關懷與愛之下完成論文。

最後，謹以此文獻給我摯愛的阿嬤與雙親。

陳冠翰　謹誌

2007 年 10 月 于交通大學工程三館 219 實驗室

目　　錄

表 目 錄

圖 目 錄

Chapter 1

Introduction

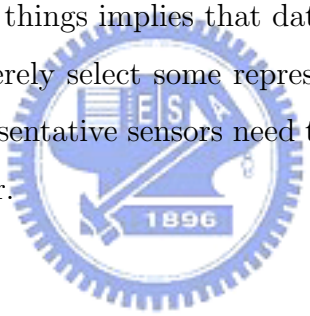
1.1 Overview

Recent advances in wireless networking and electronics have enlightened the development of wireless sensor node. Such nodes are small-scale in size, low-power in communication and low-cost in production. Each sensor node would be able to sense ambient states in the environment, process the collected data by simple functions and transmit the results to neighboring nodes. A wireless sensor network (abbreviated as WSN) is made up of a large amount of wireless sensor nodes, which randomly or arbitrarily scattered over the sensing field and cooperated to accomplish specific purposes. Since sensor nodes are restricted by the transmission range, they route data to the *sink* by a multi-hop manner that surrounding sensors help the source sensor to relay data. *Sink* is a base station to collect and deal with the raw data. Sink may communicate with Internet and answer the current states about the sensing field. Therefore, WSN can be deployed in some unreachable terrains or dangerous circumstances, such as calamity monitoring (forest fire), firsthand information in battlefield, and the care of medical treatment.

Nowadays, there are many existing issues in wireless sensor networks. One of the most common concerns is energy efficiency. As sensor nodes only have limited source of energy, power management and power conservation take on a very important role. In-network query processing with data aggregation is one way to prolong the lifetime of sensor nodes in many surveillance applications. In such environment, the raw data sensed by the sensor nodes are ordinarily processed within the network and only aggregated information would return to the sink. Certain nodes

which collect and process the other sensors' information are called *aggregated nodes*. These nodes are usually designed for higher efficiency in data gathering and computing power. They transform raw data into serviceable information by using functions such as suppression, MIN, MAX, SUM, and AVERAGE. Furthermore, since the transmission cost is higher than computation, sensor nodes are usually organized into clusters. The members of a cluster elect a representative node, called *clusterhead*, to manage the whole cluster. Aggregated nodes are similar to clusterheads in some network applications. It conserves a great deal of power consumption for sensor networks because a part of nodes pre-process and compress the data information within the network.

In WSN, data correlation is usually related to different spatial districts. Sensors observe correlated readings when they are placed into a geographical environment which is appearing the same events. For instance, if we densely deployed the sensors over many classrooms and monitored the variations of temperature, we would discover that the sensors in the same classroom detected similar temperature readings simply because the air conditioner is shared in the same spatial region. The appearance of things implies that data correlation is proportional to spatial correlation. Therefore, we can merely select some representative sensors to stand for others in the same region. Then, only representative sensors need to report the data so the overall energy consumption would be much lower.



1.2 Objective

In this thesis, we address the problem of data gathering over a highly spatial-correlated region, which means adjacent sensors have the similar readings. We then propose an energy-efficient data gathering (EDG) scheme to partition the network into such a set of clusters, which have monitored similar phenomena. Each cluster is represented by a clusterhead that is responsible for reporting the current sensing reading to the sink periodically. The establishment of clusters is via a query from the sink to the whole networks accompanied with an user-tolerable threshold. The sensing readings are accepted by the users when the readings are within the threshold with respect to the corresponding clusterheads.

After establishing the set of clusters, each cluster reports information to the sink. EDG utilizes the information for merging and reducing the number of clusters which have similar

readings in vicinity for the sake of preserving energy consumption. The conditions of combining the clusters are based on the geographical distributions and the relations with their readings. In the process of dealing with the combinations, we derive an interesting problem called infer-graph set problem which tends to select the minimal number of representative clusters. We propose a greedy-based algorithm to acquire the near-optimal solution in our thesis.

In addition to the greedy algorithm, we also introduce the concept of safe region for sensor nodes to avoid the overhead of rebuilding the clusters. In WSNs, users usually pay attention to dramatic data changes rather than slight variances in states or numbers such as temperature or humidity. Moreover, the changes may lead to sensors out of the restriction of a cluster and have to re-establish the clusters, especially for the clusterhead nodes. To avoid this situation, the safe region allow each node (even clusterhead or member node) a floating range. That means sensors will not leave their clusters unless the readings are beyond the pre-determined range. It economizes a great deal of power consumption on unnecessary cost of rebuilding clusters.

1.3 Contribution

Overall, we summarize the contributions of this thesis as follows:

1. We propose a mechanism to collect sensor nodes into clusters by exploiting the spatial correlation of sensing readings.

2. We present an algorithm in the sink that merge some clusters with similar readings underlying the user-tolerable threshold, and we offer an heuristic algorithm to select minimal representative clusters.

3. We further propose the concept of safe region so that each sensor node will not easily break the condition in a cluster. It reduces the overhead of rebuilding clusters.

1.4 Organization

The remainder of this thesis is organized as follows. First, Section 2 presents related works that has been done previously. The system architecture and the proof are then described in Section 3.

Section 4 evaluates our scheme and shows the experimental results. Finally, Section 5 concludes this thesis.

Chapter 2

Related Work

In this chapter, we survey several in-network aggregation routing protocols in WSN. They are classified into two categories, which are *flat (data-centric routing*, *hierarchical routing*. Different types of routing protocols are beneficial to dissimilar applications. In flat routing, sink sends queries for data to certain regions and the neighbor nodes in the communication range report the answers to the sink if they got the sources that sink wanted. $SPIN$ [4] and DD [5] are typical routing manners in this category. Hierarchical routing is suitable for the large-scale sensing environment. We further subdivide hierarchical routing into two categories. One is general hierarchical routing, the other is spatial-correlated hierarchical routing. General hierarchical routing distinguishes sensors into multi-tiers that sensors deal with diverse duties in each layer. Some of general hierarchical routings combine data aggregation and fusion in the cluster and only clusterheads(representative nodes) transmit messages to the sink in order to decrease the energy usage, such as $TTDD$ [7], $LEACH$ [3]. The others permit approximated answers. They are designed to be responsive to sudden changes and allowing continuous responses to increase the practicability of routing tree, such as $TEEN$ [8] and $APTEEN$ [9]. As to spatial-correlated hierarchical routing, it integrates the features of the former routing approaches. A user requests a query from a single node that establishes the forwarding tree and identifies the cluster through distributed algorithms in real-time, such as DSC [10], $SnapshotQueries$[6] and CAG [12]. In the following sections, we briefly sketched these routing protocols and illustrated one of each category particularly.

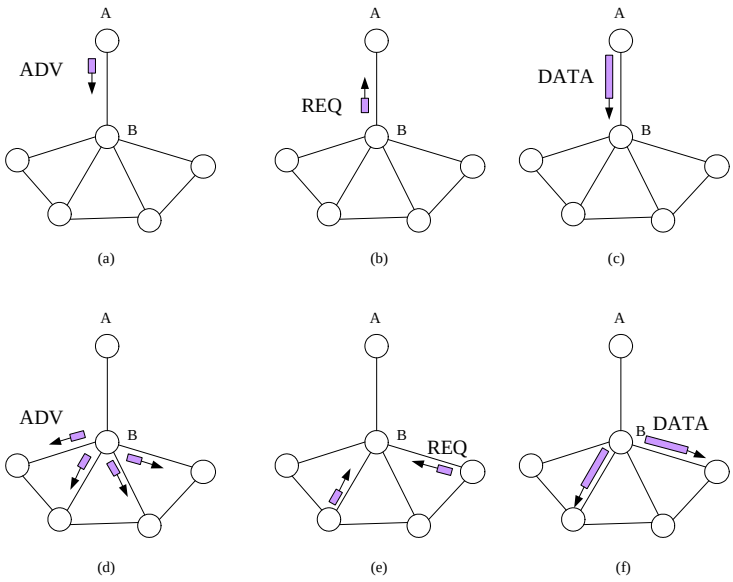


Figure 2.1: SPIN protocol
Node A advertises its data to node B (a). Node B responds by sending a request to node A (b). After receiving the requested data (c), node B then sends out advertisement to its neighbors (d), who in turn send request back to B (e-f).

2.1 Flat Routing

Sensor Protocols for Information via Negotiation (abbreviated as SPIN) [4] is a data-centric routing structure which disseminates information from each node to the whole network. A source node first broadcasts its neighbor nodes with an advertising packet(ADV), called meta-data, which shortly describes the skeleton that the source node got. When a neighbor node receives the advertisement, it responds a requested packet(REQ) to the source node in the cause of retrieving the data. Eventually, the source node sends out the completed data to the requested neighbor node. The illustration is shown as Fig. 2.1. Unlike the classic problems in flooding, SPIN utilizes meta-data with small memory size to avoid the shortcomings of flooding, such as sensing areas overlapping and redundant information passing. Nevertheless, SPIN cannot guarantee the delivery of interested data. If a node is interested in the data from the other source node and the relayed nodes between source and destination are not interested in that data, the destination will never acquire the data.

Another flat routing approach is Directed Diffusion (abbreviated as DD) [5].The idea aims at diffusing data through sensor nodes which creating gradients of information in their respective

neighbor nodes. The sink node broadcasts interests for requesting data. If gradients and interests match, information paths are established between sink and sources for the sake of diminishing communication costs by aggregating data on the way. However, since Directed Diffusion focuses on query-driven model, it will work inefficiently in continuous response-based applications, such as environmental monitoring.

2.2 Hierarchical Routing

2.2.1 General Routing

LEACH [3], also known as Low-Energy Adaptive Clustering Hierarchy, is one of the first hierarchical routing methods for wireless sensor networks. The idea of LEACH is to group together the sensor nodes by exploiting the received signal strength and randomly change local clusterheads as routers, so the transmission cost can be spread to all the sensors in the cluster. Under LEACH mechanism, each node chooses a random number between 0 and 1. The node turns into a clusterhead if the number is less than the following threshold:

$$T(n) = \begin{cases} \frac{p}{1-p*(r \bmod \frac{1}{p})} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases}$$

where p is the desired precentage of clusterheads, r is the current round, and G is the set of nodes that have not been clusterheads in the last $\frac{1}{p}$ rounds. Only five percent of the total sensor nodes became chusterheads. It will save energy because transmissions are made solely by clusterheads and clusterheads take turns to balance the energy cost. However, LEACH has an irrational assumption that each node can transmit immediately to the sink.

Threshold sensitive Energy Efficient sensor Network protocol (abbreviated as TEEN) [8] and Adaptive Threshold sensitive Energy Efficient sensor Network protocol (abbreviated as APTEEN) [9] are other hierarchical protocols which utilizing thresholds to decrease the amount of transmissions. Fig. 2.2 depicted the construction, which is redrawn from [8]. TEEN forms nearby nodes as clusters which are separated into several levels. The nodes in each cluster report back to clusterhead only when the sensed reading fall above the hard threshold, and clusterheads further reduce the number of transmissions when changes by a given soft threshold. Nevertheless,

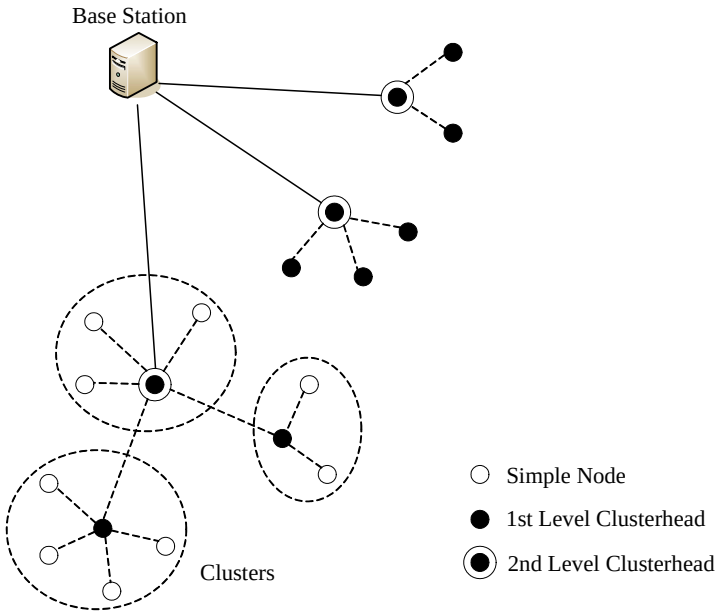


Figure 2.2: Hierarchical clustering in TEEN and APTEEN

TEEN does not support periodic reports since data only transmits while it broke the thresholds. In view of this, APTEEN addresses both periodic data collection and reporting of time-critical events. TEEN and APTEEN guide a concept through permitting approximate results instead of exact answers.

2.2.2 Spatial-Correlated Routing

Distributed Spatial Clustering in Sensor Networks(DSC) [10] addresses the problem of discovering spatial relationship in sensor data through the identification of clusters. DSC proposed a δ-cluster structure that the sensors were partitioned into cluster according to the δ. It broadcasts a packets from a root node to its neighbors with a threshold δ, if the feature distance between two nodes is less than $\frac{\delta}{2}$, the root node includes the neighbor in its cluster. Since the distance between the feature value of any node in the cluster and the root feature is at most $\frac{\delta}{2}$, triangle inequality ensures that the distance between the feature values of any two nodes in the cluster is at most δ. Fig. 2.3 is an example that redrawn from [10].

Snapshot Queries [6] aims at determining the representative nodes among groups of sensors with similar observation. Snapshot exchanges data information with neighbors 1-hop away and predicts the similarity by employing a linear regression model for the sake of establishing clusters.

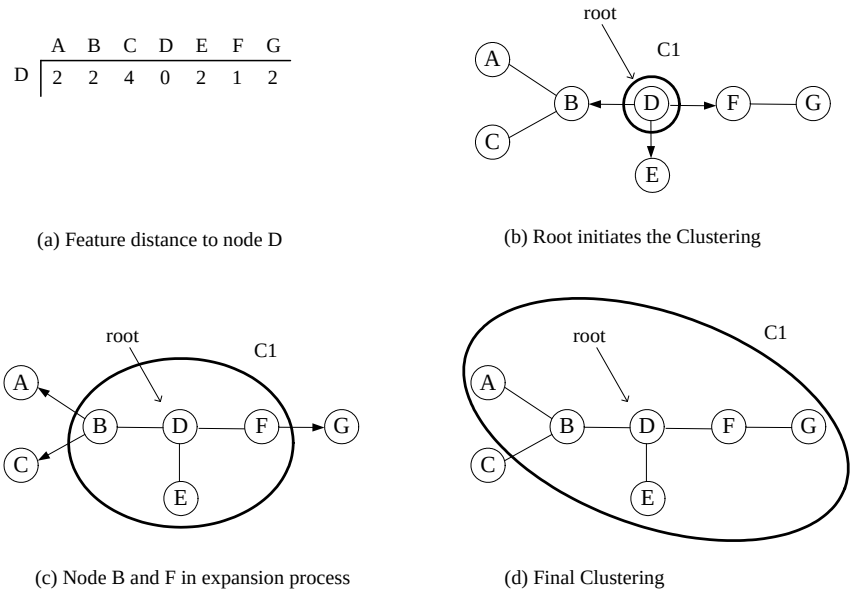


Figure 2.3: DSC protocol
For $\delta = 6$, the feature distance of every other node to sentinel node D are shown in (a). The expansion of cluster C1 starts as shown in (b), continues as shown in (c), and terminates as shown in (d).

The sensor nodes make use of the distributed algorithms to maintain and elect a smaller set of representative nodes in a localized fashion using a small(up to six) number of messages per node.

Exploiting Spatial Correlation Towards an Energy Efficient Clustered AGgregation Technique (abbreviated as CAG) [12] allows approximate results to aggregate queries by utilizing the spatial correlation of sensor data and save energy. The CAG algorithm operates in two phase: query and response. The former phase forms clusters by broadcasting a user-specified error threshold τ on the way of establishing the forwarding tree. Each clusterhead can choose its own members depends on the sensed reading of each clusterhead and threshold τ. Once all the nodes receive the query packets, the latter phase starts. Only a select of clusterheads transmit to the sink where they represented. Considering the phenomena of spatial correlation in sensor networks, CAG further improve the utilization of energy.

2.3 Comparison

Table 2.1 summarizes the classification of the above protocols. We further included two metrics in the table. One is to see if the protocol has the ability of aggregation, since it is an significant

Routing protocol	Flat	Hierarchical	Spatial correlation	Aggregation	Approximate answer
SPIN	Yes	No	No	Yes	No
Directed Diffusion	Yes	No	No	Yes	No
LEACH	No	Yes	No	Yes	No
TEEN and APTEEN	No	Yes	No	Yes	Yes
Snapshot Queries	No	Yes	Yes	Yes	Yes
DSC	No	Yes	Yes	Yes	Yes
CAG	No	Yes	Yes	Yes	Yes

Table 2.1: Comparisons of related works

index for power consumption. The other is to examine whether the protocol accepts approximate answer or not, which allowed the protocol to deal with data flexibly. We can observe a trend that protocols exploiting spatial correlation are all capable of accepting approximate answers and utilizing data aggregation. As facing increasingly difficult of sensing environments, more techniques are provided for routing protocols to handle the complexity.

Chapter 3

EDG: Energy-Efficient Data Gathering Scheme

An overview of scheme EDG is described in Section 3.1. We then subdivide EDG into three procedures. First, Section 3.2 introduces how to establish and determine the clusters. Then we present an algorithm in the sink to regroup the clusters into fewer, larger representative clusters in Section 3.3. Finally, Section 3.4 provides each sensor node a safe region to avoid rebuilding the network topology frequently.

3.1 System Overview

Fig. 3.1 illustrates the procedures of scheme EDG. A number of sensors are deployed in a sensing field to monitor the environment. Since our scheme is based on an interactive query system and accepts approximate answers, users request a query from the sink with a user-tolerable

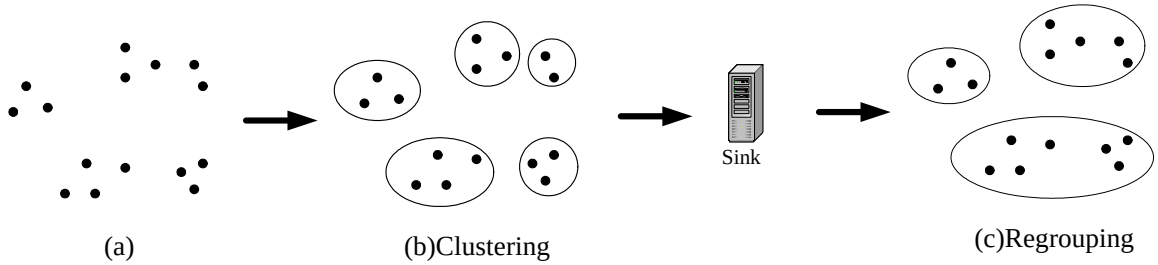


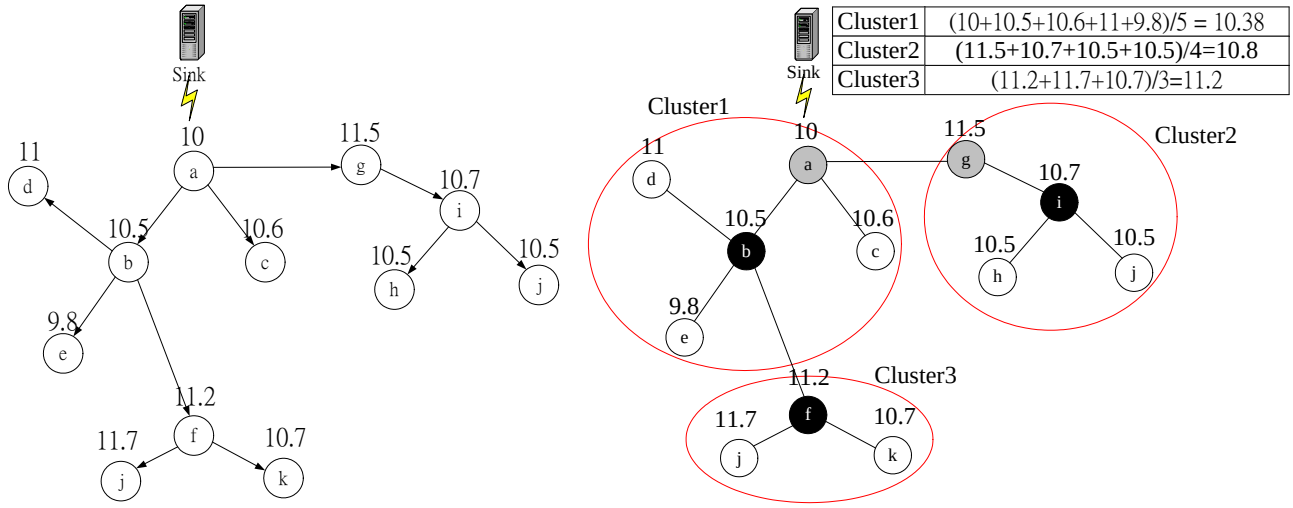
Figure 3.1: System Architecture

threshold. Sink then broadcasts the messages to the whole networks to build up the first version of clusters based on the threshold. The result is shown as Fig. 3.1b after clustering procedure. Then in regrouping procedure, each cluster reports some information to sink. Sink calculates the representative relations of every two clusters according to their reported values. To obtain the number of representative clusters, we model it as an infer-graph set problem. In addition, we devise a heuristic algorithm, i.e., algorithm *Regrouping*, to acquire the approximate optimal solution of the number of representative clusters. Then sink again broadcasts the adjusted messages to merge networks into fewer clusters, as shown in Fig. 3.1c. In the last procedure, we propose each sensor a safe region to float the reading within the range. It conserves a large amount of power consumption since sensors do not reconstruct the topology on lightly reading changes.

3.2 Clustering

First, when a user makes a query from the sink, the query is defined as $< F, \epsilon >$. F (Feature) is the feature value, which the user inquires about a region, like Humidity, Temperature and so forth. ϵ (error threshold) is the user-provided threshold which the user estimates. After the sink receives the query the user makes, it starts to form the former phase clustering. The sink broadcasts a query packet and the ingredients are Q=$< F, TCID, TCR, MyID, \epsilon >$. F (feature) is the user-specified feature value, TCID (Temporary Clusterhead ID) represents the identification of the temporary clusterhead, TCR (Temporary Clusterhead Reading) is the reading of the temporary clusterhead, MyID is the identification of the sensor itself used for transmitting the result and ‘is the error threshold of estimated answers.

After a sensor near the sink receives a packet, it will confirm the feature and then retrieve its Local Reading (LR, Local Reading) to compare it with TCR of Q. If LR falls between $[TCR - \epsilon, TCR + \epsilon]$, the sensor will be taken as a clusterhead and broadcast the same Q to the neighbor sensors. If LR is not within TCR, the sensor needs to establish another new cluster and becomes its Temporary Clusterhead, in the meanwhile broadcasting a new query packet Q’=$< F, TCID', TCR', MyID, \epsilon >$. TCID’ is the identification of its sensor and TCR’ is the LR of the sensor. It’ll be broadcasted in this way until all the sensors in the sensor field receive



(a) Before (b) After

Figure 3.2: Example of clustering with user-tolerable threshold $\epsilon = 1$

a query packet.

After the sensors receive the packets, they'll transmit a response packet, which can be defined as R=$< MenberID, LR >$. Member ID is the identification of the sensor and LR is the local reading of the sensor. After a temporary clusterhead aggregates all the information from its cluster members, it has to reselect a new clusterhead. The temporary clusterhead averages all the local readings and gets an average value. The most approximate member is the new clusterhead, which will transmit the result to the sink. The main purpose is to keep the clusterhead reading in the middle value of all the cluster members and to decrease the possibility of reforming a new cluster for a minor reading floating among some marginal members, which will be explained in Section 3.4. When all the new clusterheads are determined, later on only they aggregate and transmit the information in the whole sensor field. The reading of each clusterhead represents the readings of all the cluster members to achieve the energy saving.

The algorithmic form of algorithm Clustering is as follows.

Algorithm Query(do in each node)

1: **if** $(TCR - \epsilon) \leq LR < (TCR + \epsilon)$ **then**
2: clsuterID = $TCID$
3: clusterhead = false
4: broadcast query Q with same $TCID$, TCR
5: **else**
6: $TCID$ = i.getID
7: TCR = i.getReading

8: broadcast query Q with new $TCID$, TCR
9: **end if**

Algorithm Re-electing Clusterhead and Response

1: **for** each member node **do**
2: Report current sensing reading to the temporary clusterhead.
3: **end for**
4: **for** each temporary clusterhead node **do**
5: **if** collect all the member nodes' readings **then**
6: average the members' readings as avg.
7: take the member whose reading is most close to the avg as $NewClusterHead(NCH)$.
8: notify all members in the cluster.
9: **end if**
10: **end for**
11: NCHs response the current readings periodically

Take Fig. 3.2 for example, when a user wants to know the reading of the sensor field and makes a request in the sink: $< F, \epsilon >=< Temperature, 1 >$, the sink initiates the package and then node a transmits the query to its neighbor nodes b, c, g. The query is Q=$< Temperature, IDa, 10, IDa, 1 >$. When node b receives the query, it will check if its local reading falls within [TCR - ϵ, TCR + ϵ]([9A11] in this example) Because the local reading of node b falls within the range, node b becomes a member of node a and belongs to the same cluster and then keeps broadcasting the same query to its neighbors.(only changing MyID to IDb) If node g receives the query from node a, its local reading is exceeding 11.5, so it would become a new clusterhead itself. When it goes on broadcasting, it will revise the query: Q=$< Temperature, IDg, 11.5, IDg, 1 >$. It will keep broadcasting this until all the nodes in the sensor field receive the query. In this example, the readings of node b, c, d and e are within the error range and thus they all become in the same cluster. The way to select a clusterhead is the first node whose sensor reading is outside of the tolerable error range. Clusterhead selection policy is the first node becomes a clusterhead if its sensor reading is outside of the tolerable error range. Therefore, node g and node f in this example form new clusters.

When the temporary clusterheads, node a, g and f, are all set, it will reselect new clusterheads. For cluster 1, temporary clusterhead node a collects all the readings from its members, it equalizes those readings, 10.38 and reselects a node whose reading is the closest to be a new clusterhead (NCH), node b, which will send back information in cluster1 from then on. In cluster 2, the most closest to the average (10.8) is node I, NCH. Because the average is equal to temporary

clusterhead in cluster 3, node f is still the clusterhead. From above, we will finish the clustering phase and reselect new clusters and clusterheads.

3.3 Regrouping

3.3.1 Conditions of Regrouping

After we sorted each cluster, we are thinking if it can be more energy efficient in some circumstances. Therefore, regrouping is developed and regarded as the main idea in this procedure. In the sensor field of the highly spatial correlation, the sensing data are extremely close to each other due to their spatial correlation. If there is no characteristic frequency change, we prefer each cluster to send few messages to the sink after clustering. After the sink collects all the information from clusters and then analyzes and integrates it, the sink can combine some clusters into a bigger one. By doing that, it can save some extra energy consumption in the next report. Here is the detailed explanation for this process. After establishing clustering procedure, the clusterhead(CH) of each cluster transmits the following three values to the sink. We take one cluster C_i for example.

1. Current reading of C_i, represented as V_i.
2. The absolute value of maximum difference between all members and CH in C_i, represented as $|MaxDif_i|$.
3. A ID list of clusters which neighbors C_i, represented as $Neighbor_i$.

After the sink collects the three values from all the clusters, it contrasts the two conditions of every two clusters. If the two conditions are accordable, it means that the two clusters can be merged into one. For instance, the precondition which C_i can merge C_j is followed.

Definition 1. *Given two clusters C_i and C_j, C_i can represent C_j if:*

1. $V_j \pm |MaxDif_j| \subseteq V_i \pm \epsilon$.
2. *C_i and C_j are neighbor clusters.*

C_i(Clusterhead node)	V_i	$\|MaxDif_i\|$	$Neighbor_i$
1 (b)	10.5	0.7	2,3
2 (i)	10.7	0.8	1
3 (f)	11.2	0.5	1

Table 3.1: Data report by clusterheads

If the two conditions are accordable, it means C_i can represent C_j. Two clusters are merged into new one which is led by CH of C_i. The following is the explanation for the conditions in accordance with them. For the first condition, $V_j \pm |MaxDif_j|$ is the *member range* of C_j, which means all the member readings of C_j fall in this range; $V_i \pm \epsilon$ is the *user-tolerable range* of C_i, which indicates user can accept the readings within this range. Thus, if the member range of C_j falls within the user-tolerable range of C_i, it implies the CH of C_i can represent all the member readings of C_j under user-provided threshold. For the second condition, it exists for spatial correlation. That is only when the two clusters are spatially close, we are able to combine the two. Otherwise, it would cause the irrational situations that the two clusters far from each other in distance combine into one just because their similar readings. In accordance with the above, we consider the two clusters can be merged. After the sink works out all accordance with the above, it broadcasts the messages to the whole sensor field and informs each cluster to combine with another one to reduce the number of clusters. In a long-term perspective, this way can be more energy efficient.

In the following, we still use the example in Fig. 3.2b to elaborate the detailed process of the regrouping procedure. In the clustering procedure, we make the sensor field into three clusters and ascertain the clusterheads. In the regrouping phase, the clusterhead sends back the three values to the sink. Take node b, the clusterhead in cluster 1, for example. It sends back its own reading 10.5 and the absolute value of maximum difference 0.7, i.e. difference between critical node, node e, and the clusterhead. It also sends back its neighbor cluster ID, i.e. 2 and 3. After the sink collects all the reading from all the clusters, it will calculate to see if it can combine them into a bigger cluster. As illustration shown in Fig. 3.3a, we take cluster 1 and 2 for instance, the member range of cluster 1 is $10.5 \pm 0.7 = [9.8,11.2]$, which falls in the user-tolerable range of cluster 2, $10.7 \pm 2/2 = [9.7,11.7]$. It shows that all the sensing readings of cluster 1 can be represented by the reading of the clusterhead of cluster 2 within the user-provided threshold ± 1.

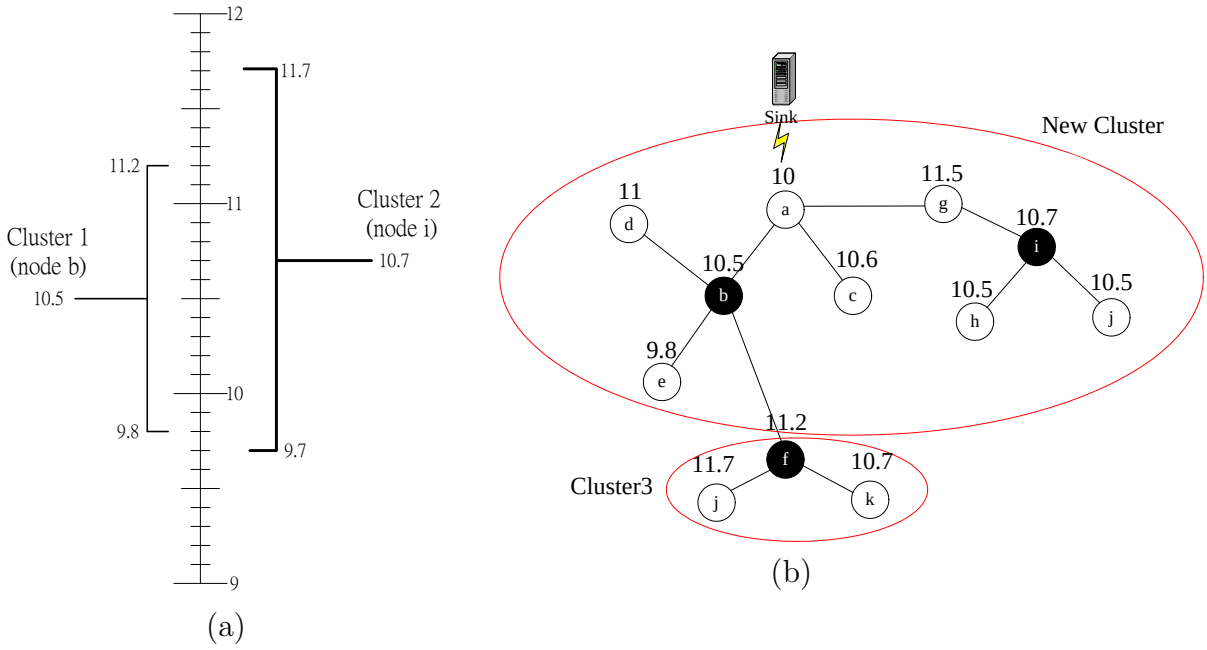


Figure 3.3: Example of regrouping

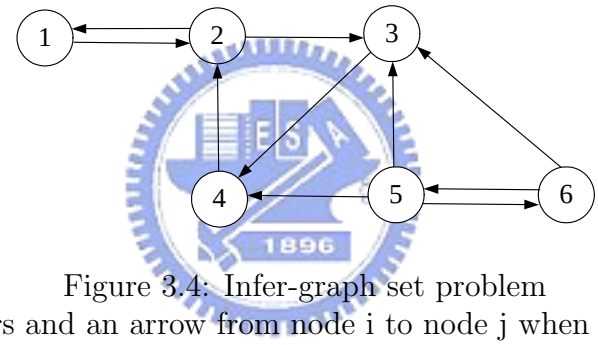


Figure 3.4: Infer-graph set problem
Nodes symbol the clusters and an arrow from node i to node j when node i can represent node j.

Moreover, cluster 1 and cluster 2 are neighbor clusters so we can conclude that cluster 1 can be represented by cluster 2. The two clusters can be merged into a bigger one which leads by node i. On the other hand, we have to calculate to see if the member range of cluster 2 (10.7 ± 0.8=[9.9,11.5]) can be included by the user-tolerable range of cluster 1 ($10.5 \pm 2/2$=[9.5,11.5]). Fortunately in this example, cluster 1 also can represent cluster 2 by node b, thus node b and node i can be the clusterhead in turn. It further promotes the utilization of the merged cluster. The final result is shown in Fig. 3.3b.

3.3.2 Infer-graph Set Problem

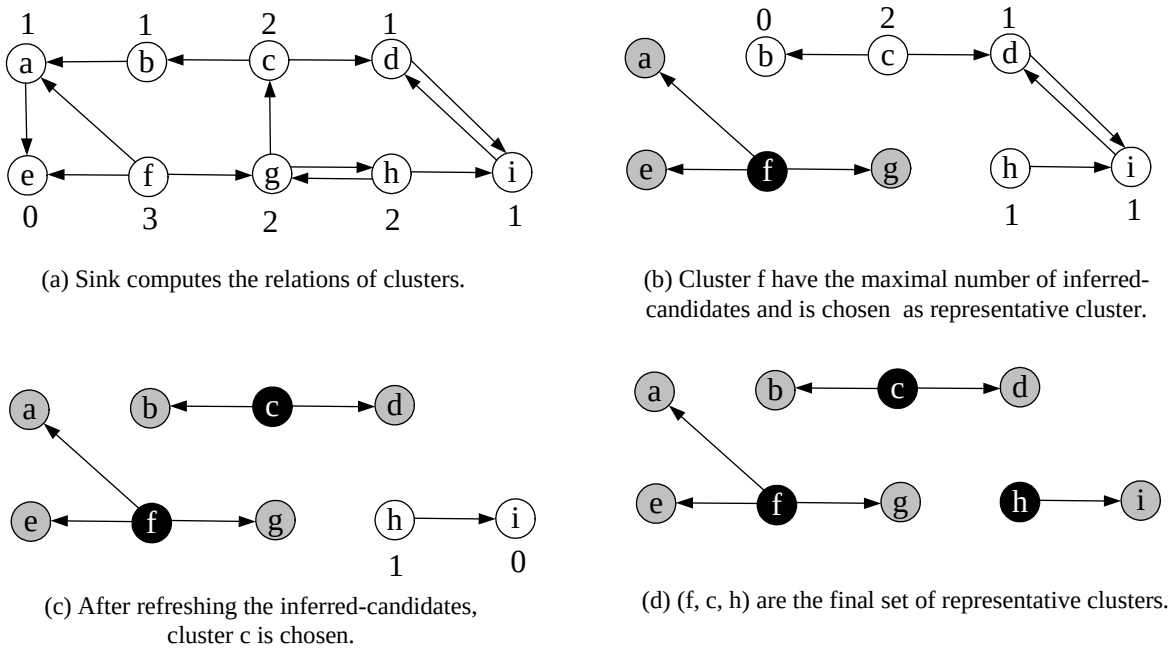
After the sink calculates all the representative relations of every two clusters, it have to determine how to choice the combinations of the new topology. As shown in Fig. 3.4, each node stands for a cluster, and the arrow from node i to node j means cluster i can represent cluster j, we can take the combination $U = (\mathbf{1}), (\mathbf{4}, 2), (\mathbf{6}, 3, 5)$ as the new topology, which each parentheses means a new merged cluster and the first item of each parentheses indicates the led cluster. However, we also can make another decision as $U' = (\mathbf{1}, 2), (\mathbf{5}, 3, 4, 6)$ with fewer representative clusters. Hence, there exists a problem to find the representative clusters, we called it infer-graph set problem, the problem can be formally defined as follows.

Definition 2. *Given a directed graph $G = (V, E)$, V is set of clusters, for arbitrary clusters $u, v \in V$, an directed edge $(u, v) \in E$ indicates cluster u can represent cluster v. An* ***infer-graph set*** *is to find a subset of clusters $V' \subseteq V$, such that each directed edge (u, v) in E is incident on at most one cluster in V'.*

We therefore propose a greedy algorithm in order to earn the local maximal benefit. Here is the algorithm for regrouping.

Algorithm Regroup Clusters by Greedy Approach
Input:
C: set of all the clusters ID
v_i: reading of cluster i, $i \in C$
$|maxdif_i|$: the absolute value of maximum difference of cluster i
ϵ: the user-tolerable threshold
Output:
R: set of representative clusters, where $R \subseteq S$

```
1: for each cluster i ∈ C do
2:    for each neighbor cluster j do
3:       if (v_j ± |maxdif_j| ⊆ v_i ± epsilon) then
4:          C_j represent C_i.
5:          i adds j to its infer-candidates u_i.
6:       end if
7:    end for
8: end for
9: while (C ≠ NULL) do
10:   Find a cluster i which has the maximum size of |u_i|.
11:   Add i to R.
12:   for each cluster j ∈ u_i do
13:      Remove j from C.
14:   end for
15: end while
```



(a) Sink computes the relations of clusters.

(b) Cluster f have the maximal number of inferred-candidates and is chosen as representative cluster.

(c) After refreshing the inferred-candidates, cluster c is chosen.

(d) (f, c, h) are the final set of representative clusters.

Figure 3.5: Example for regrouping algorithm

16: Return R for the answer.

Line 1 to line 8 explained what Section 3.3.1 have done. Sink first computes the relations between each cluster and its neighboring clusters. If C_i can represent C_j, then C_j will be added to the infer-candidates of C_i. All the clusters have calculated their infer-candidates. In the first round, sink selects the cluster who has largest number of infer-candidates, and add to the set of representative clusters R. It can be seen in line 10 and 11. From line 12 and 13, the representative cluster removes the members of the infer-candidates from the set of remaining clusters S. It iterates until there is no more cluster in S. Then we can derive a set of representative clusters from R. Fig. 3.5 illustrates an example for regrouping algorithm. The direction of an arrow means the infer-candidate of a cluster. The number near a node is the size of infer-candidates, it can also be seen as the number of out-degree of a node in the graph theorem. In the first iteration, sink select cluster f as a representative cluster since it has the largest number of infer-candidates. Then these candidates are eliminated from the set, and the remaining clusters refresh the number of infer-candidates. In the second iteration, cluster c is chosen to be the representative cluster among the remainder clusters. Finally, f, c, h is the greedy solution for this example.

After this regrouping procedure, the number of clusters would decrease because of the com-

bination. Thus, fewer clusterheads need to send back the information. In the long run, the reporting packets would be much fewer so it will lengthen the lifetime of the whole sensor network.

3.4 Safe Region

In the past, many in-network studies investigate how to effectively set up the routing tree and form the clusters for economizing the energy attrition. Unfortunately, seldom of these works center on maintaining the robustness of the clusters so that the sensors can further save power by avoiding re-establishing the clusters frequently. The longer a cluster lives, the more energy conserve for all the members in cluster. In this section, we discuss how to intensify the structure and prolong the lifetime of a cluster.

Since we focus our work on environmental monitoring, sensor nodes report the current data periodically to the clusterheads and clusterheads can do some aggregation and reduction of data. At the same time, sensing attributes such as temperature varies based on terrains or circumstances. That means the reading of a sensor node also changes over time. It brings an inevitable problem that some member nodes which had critical readings may break the ambit of cluster. These nodes are forced to leave the original cluster and try to join other available clusters. Furthermore, if a clusterhead changes the reading caused by environment, it cannot guarantee be the representative node in the future anymore. This is because the rule of electing clusterhead is choosing the node whose reading is the nearest to the mean among the members. Regardless of clusterhead or member nodes, it wastes power consumption on adjusting clusters, especially on clusterhead.

In view of this, we proposed a concept of safe region, which allots each node(even clusterhead or normal member node) a floating range. If the variance of a node drifts within the floating range, the node will be kept in the cluster with the result that enhances the persistence of the cluster. Now we discuss how to generate the floating region. As mention in clustering phase, we first find out the Temporary Clusterhead(TCH) and average the readings of all the cluster members. The movement of digging out the mean not only decided the New Clusterhead(NCH) but shifted the value of clusterhead to middle of whole cluster members. Therefore, it will move

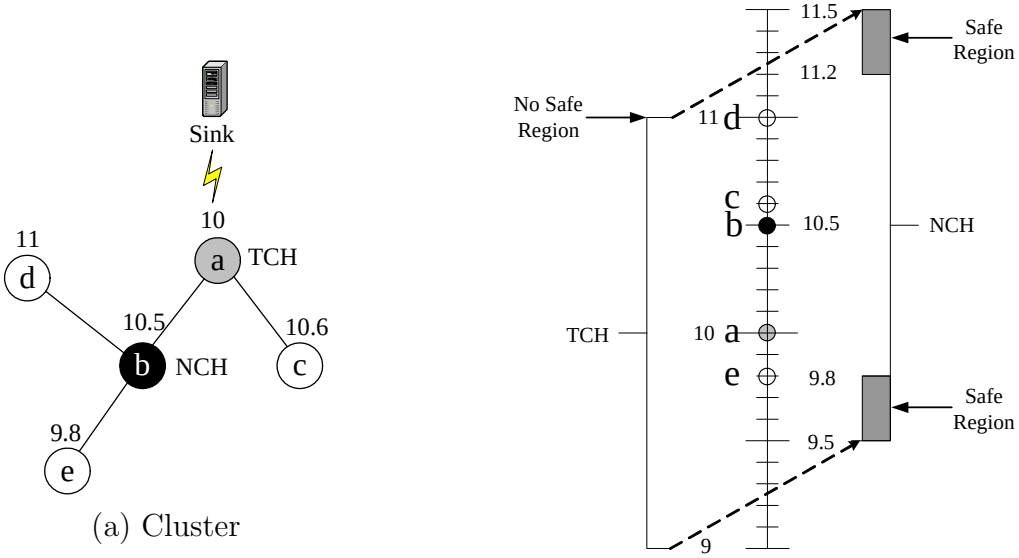


Figure 3.6: Example of safe region with user-tolerable threshold $\epsilon = 1$

out a margin of range to realize the concept of safe region. Fig. 3.6 explained this idea. Fig. 3.6b transforms Fig. 3.6a into a temperature-like chart with pointing out the corresponding nodes. We can discover that if TCH(node a) is decided to be the clusterhead, we have no safe region at all. Since it exists a critical node(node d) whose reading is fell on the boundary of TCH, which is [9,11]. After re-voting for NCH, the reading of NCH was shifted to [9.5,11.5] result in switching the critical node to node e whose reading is 9.8. Eventually, it generates a safe region, bounded in 0.3 in this example, for the cluster.

Different clusters generate distinct safe regions because the generation of a safe region is depended on NCH and critical node. The closer readings between NCH and critical node, the larger safe region sharing for a cluster. We now further divide the safe region into two sub-regions, called clusterhead floating range and membership floating range, respectively. The clusterhead floating region is allocated to clusterhead, so the membership floating range is to the members of clusters. We prefer to allow clusterheads wider floating range since the changes of the clusterheads may result in extremely energy cost, such as cluster recovering. Member nodes, however, got narrower membership floating range because they will not destroy the completeness of the clusters. In Fig. 3.6b, we got a safe region with 0.3 degrees, we might allocate NCH(node b) a clusterhead floating range with 0.2 and members a membership floating range with 0.1

degrees, respectively. This phase diminishes the possibilities of a node to switch between the clusters, thus retrenches the energy consumption.

Chapter 4

Performance Evaluation

4.1 Simulation Model

We developed a network simulator based on JSIM [13] to generate the queries and the sensors associated with their readings. Our experiments are conducted on network sizes ranging with 600 sensor nodes in a 500m × 500m two-dimensional sensing field. The field is divided into 20m × 20m grid cells. Users request the queries from the sink at point (0,0). The transmission range of each sensor node is set to 30m. The simulation time is 600 seconds and sensors report data every 10 seconds. Sensing reading of each node is generated with a uniform probability distribution from 16 to 24. Packets are divided into two categories, which are data packets and broadcast packets. The energy consumption model refers to [11]. Table 4.1 summarizes the system parameters and setting. Be convenient to narrate, we name our scheme as EDG (Energy-efficient Data Gathering) for the rest experiments.

4.2 Impact of Simulation Time

In Fig. 4.1, we compare the number of bytes of CAG and EDG approaches. The total numbers of sensor nodes are 600. Nodes report the readings in the sensing range every 10 seconds periodically. For CAG and EDG approaches, the user-tolerable threshold is set to be 0.1 with respect to corresponding clusterheads. CAG proposes an algorithm to build up the message-forwarding tree and generate the clusters in the same time. It decreases the number of bytes because only

Parameter	Setting
Sensing field size	500 m × 500 m
Number of nodes	600
Report rate	10 sec
Transmission range	30 m
Sensing Reading range	16 - 24
Threshold distance (d_0)	75 m
E_{elec}	50 nJ
ε_{fs}	10 $pJ/bit/m^2$
ε_{amp}	0.0013 $pJ/bit/m^4$
Data packet size	100 $bytes$
Broadcast packet size	25 $bytes$
Packet header size	25 $bytes$
Initial energy	2 J
Simulation Time	600 sec

Table 4.1: System parameters

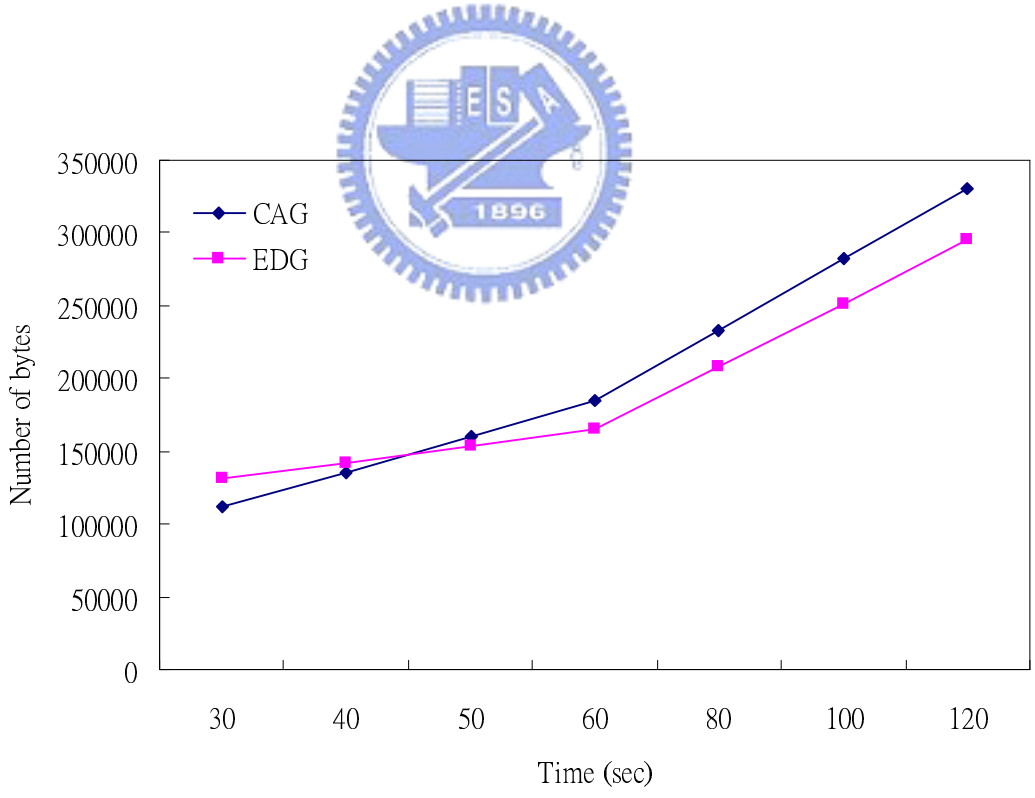


Figure 4.1: Number of bytes vs simulation time

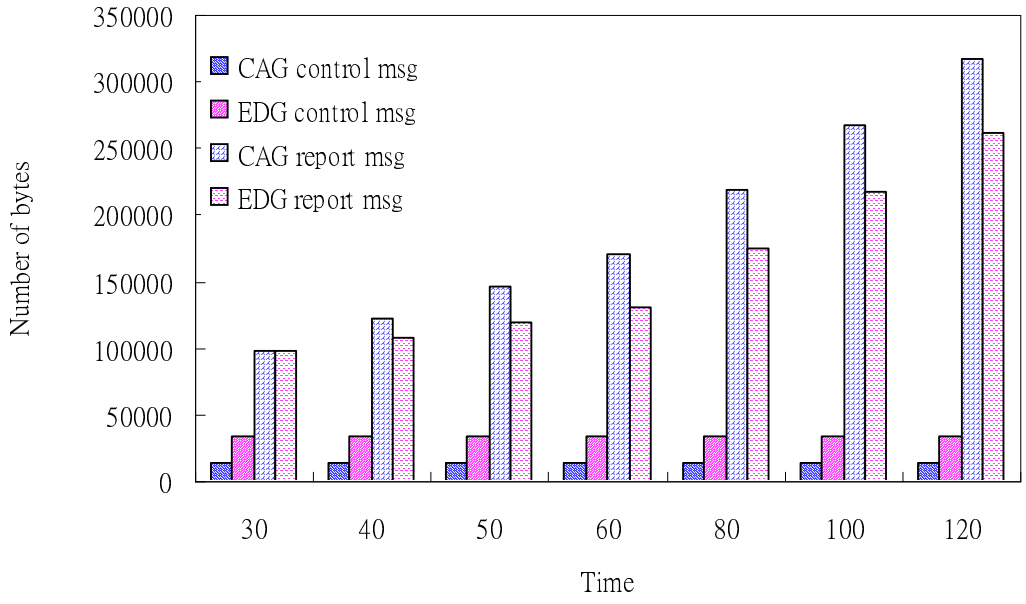


Figure 4.2: Number of alive sensors vs simulation time

clusterheads have to report the information. However, EDG not only does what CAG has done but also improves the CAG algorithm by reducing the number of clusterheads and offering the sensors the structure of safe region, so EDG has less total messages than CAG. However in 30 to 45 seconds, EDG has the overheads for waiting the regrouping messages and broadcasts adjusting messages from the sink to all the sensors for new network topology. Fig. 4.2 shows the details of total messages. We separate messages into control messages and report messages. Control messages are responsible for establishing forwarding tree and clusters, regrouping clusters in EDG approach. Periodical data reports are included by report messages. At 30 seconds, EDG has more control messages than CAG because EDG needs to exchange messages between local clusters and sink. And EDG has the same report messages due to sink not broadcast clusters combinations yet. When time increased, the number of EDG's report messages decrease caused by regrouping, and the number of control messages are the same. So in long-term viewpoint, EDG still outperforms CAG by 11 percent of the number of bytes.

We now compare the number of alive sensors with time series of different approaches. We deploy 600 sensors in sensing field. In the Naive approach, all nodes in sensing field report their readings to the sink periodically. As shown in Fig.4.3, the number of alive sensors in

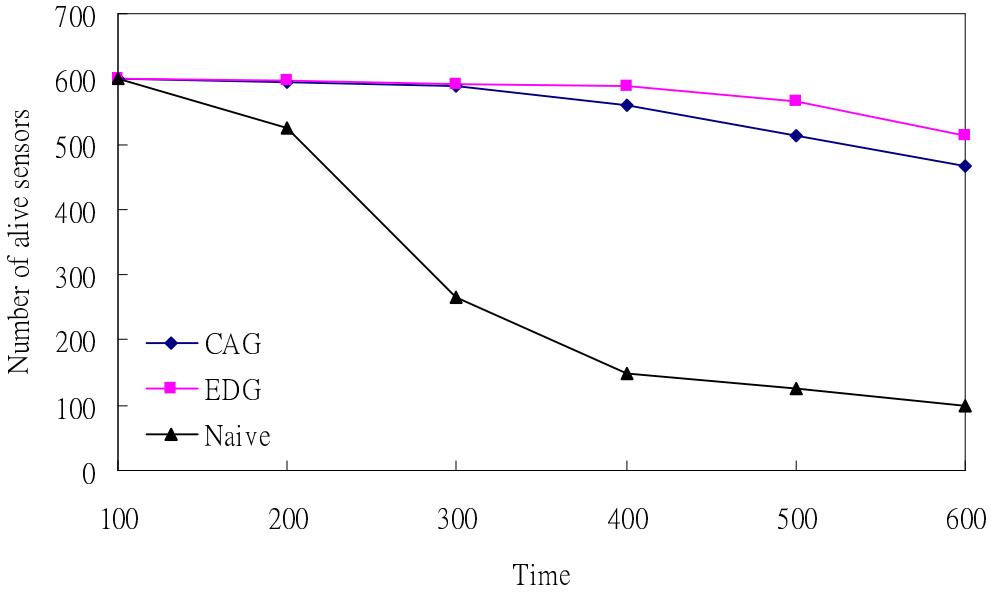


Figure 4.3: Number of alive sensors vs simulation time

Naive approach died rapidly especially from 200 seconds to 400 seconds since every sensor has to report the reading. In both CAG and EDG, they are cluster-based approaches so that the member sensors in a cluster are representative of exactly one clusterhead. Thus, the number of alive sensors diminishes slightly since only clusterheads report the readings. Moreover, EDG reduces the number of clusterheads than CAG, so sensors on the path to the sink of these reducing clusterheads have no longer to forward the messages. In the end we can see that the number of alive sensors in EDG is more than the number of alive sensors in CAG.

4.3 Impact of Sensor Number

To measure the scalability between CAG and EDG, we change the number of sensor nodes in the simulated network from 100, 200, 300, 400, 500 to 600. We vary the size of the simulated area according to keep a fixed node density. Fig. 4.4 depicts the relation between the number of clusterheads and the network size. We see the superior performance of our EDG approach. This is because EDG merged the clusters at the regrouping procedure. In CAG, some neighbor clusters which have the similar readings cannot integrate with each other since they have critical

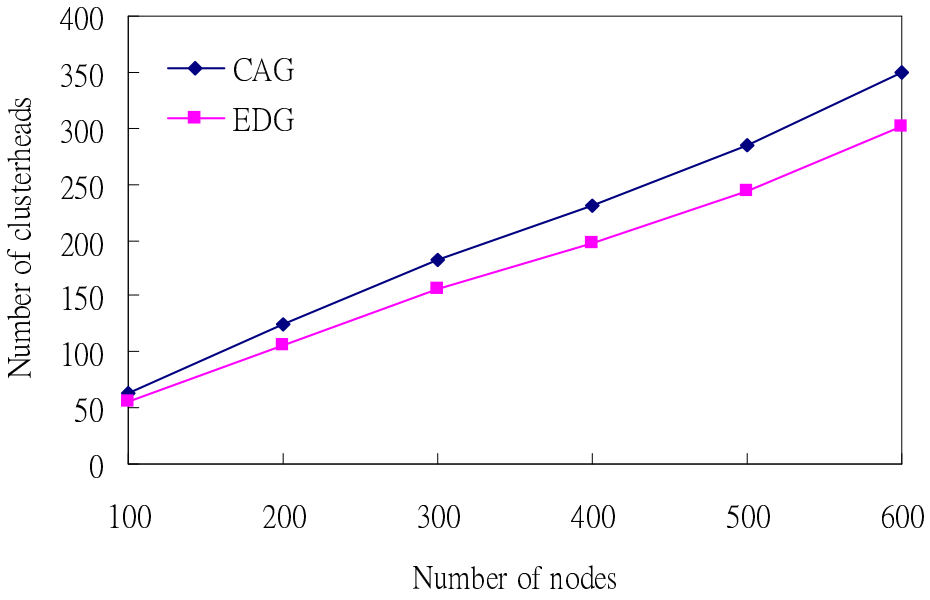


Figure 4.4: Number of clusterheads vs sensor nodes

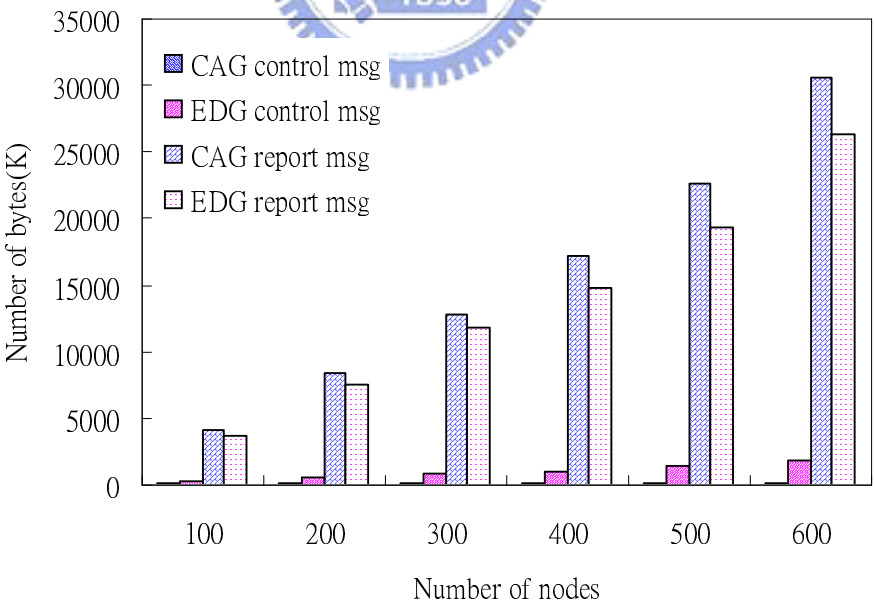


Figure 4.5: Number of messages vs sensor nodes

point between them. However, EDG solve the situation by exchanging some messages between sink and clusters to reduce the number of clusters.

Fig. 4.5 shows the number of bytes count for different number of nodes. EDG outperforms CAG in all cases. We also can discover as the network size enlarged, the number of messages of CAG raised heavily where EDG raised lightly. There are two reasons to explain this situation. One is that EDG has fewer number of clusterheads so less messages are send. The other reason is when the network size enlarged, the average distance from a sensor to sink increased. It leads more cost to transmit messages. If a clusterhead is far from sink and is merged by EDG approach, it can save a great deal of messages than CAG. This is why the message overheads of CAG is outstandingly higher than EDG.

4.4 Impact of User-tolerable Threshold

The following experiments investigate the variations of user-tolerable threshold. We deployed sensors in the sensing field and report rate is set to be 10 seconds in 200 seconds total simulation time. Fig. 4.6 and Fig. 4.7 display the number of clusterheads and messages of EDG normalized by that of CAG under the different thresholds, respectively. As shown in Fig. 4.6, when threshold is set to 0, that means users do not permit any errors for reporting answers. Every sensor is identical to a clusterhead to report exactly what it sensed. In this case, CAG and EDG are retrograded as same as Naive approach. With increasing the value of threshold, EDG reduces 15 percent of the number of clusterheads as the threshold is 0.15. In this best case, EDG diminishes 49 clusterheads than CAG. An interesting result is that the number of clusterheads of EDG is closing to CAG again when threshold is increasing. This is because the condition to form a cluster became loosed, a cluster can include more members thus the number of clusters get lessened. Considering an extreme case, all the sensors are included in one cluster because the threshold is too big, then EDG is hard to merge clusters anymore.

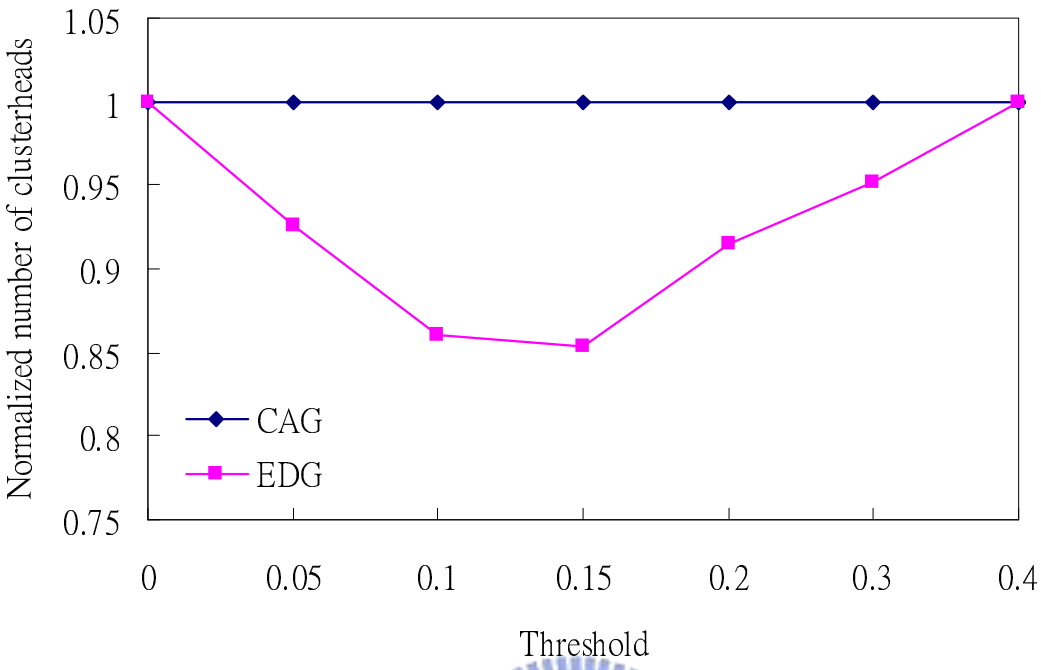


Figure 4.6: Normalized number of clusterheads vs user-tolerable threshold

4.5 Impact of Changed Events

Monitoring phenomena usually vary their features accompanied with time series so that sensors detect different sensing readings. We then evaluate the impact of the number of changed events. In Fig. 4.8, we examine the overheads for different ratios of changed events over total 600 sensors. A changed event means that a sensor node varies its sensing reading. The readings are changed by uniform distribution between a half of user-tolerable threshold with respect to the corresponding clusterheads. The number of bytes count of EDG is less than CAG all the time. By increasing the ratio of changed events, the benefit of safe region in EDG approach appears obviously. This is because EDG allows each sensor a floating range to avoid rebuilding the clusters where CAG does not have. It shows that EDG is more suitable for dynamic sensing environments.

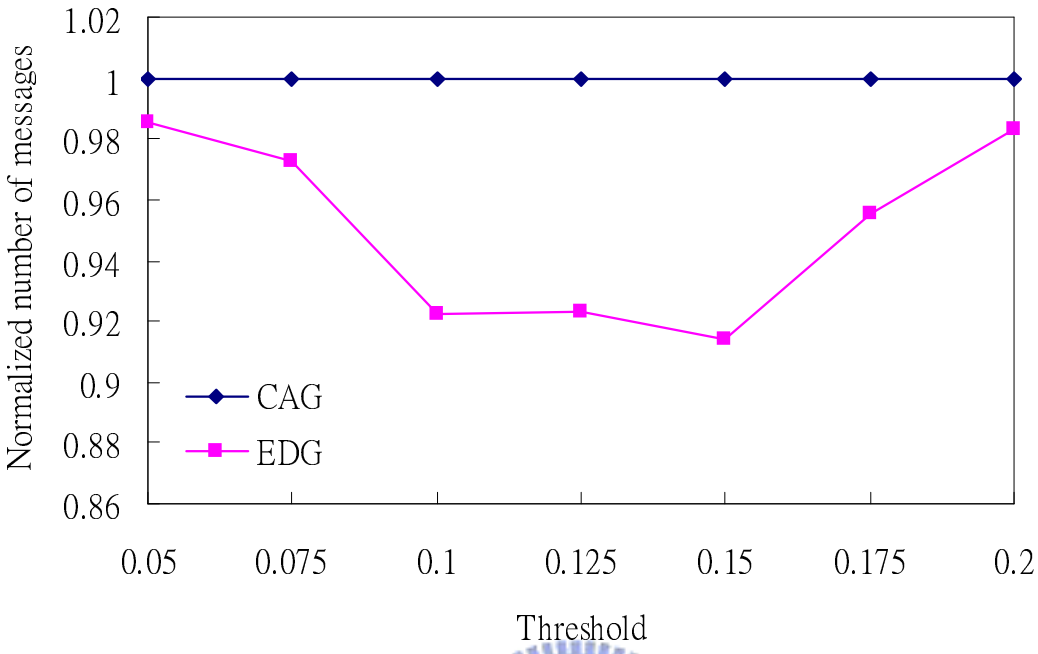


Figure 4.7: Normalized number of messages vs user-tolerable threshold

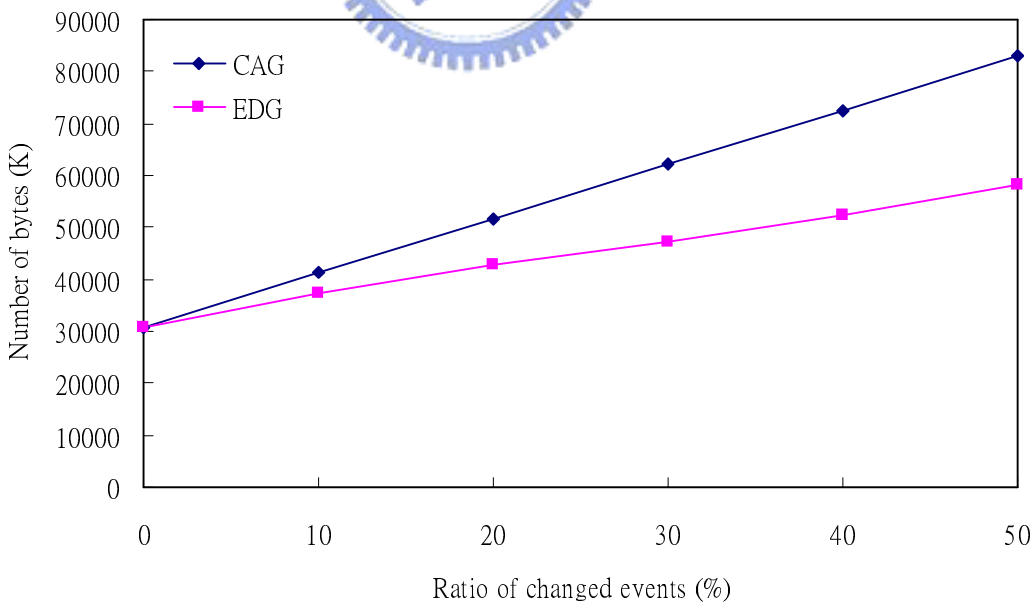


Figure 4.8: Impact of changed events

Chapter 5

Conclusion

In this thesis, we considered the problem of approximate query processing over spatial clustering in sensor networks. We formed sensors into clusters which were bounded by an user-tolerable threshold. We further merged clusters and found an innovative issue called infer-graph set problem. We then designed an heuristic algorithm to devise near-optimal solution. Moreover, the safe region robusts the persistence of clusters without losing the precision. Our experiments showed that EDG outperforms CAG 11 percent of bytes count and 15 percent of clusterheads and EDG was suited in dynamic sensing environments. We concluded that our scheme effected better energy usage of sensors and prolonged the lifetime of sensor networks.

Bibliography

[1] J. Gray, A. Bosworth, A. Layman and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery,* 1(1):29-53, 1997.

[2] H. Gupta, V. Navda, S. R. Das and V. Chowdhary. Efficient Gathering of Correlated Data in sensor Networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc),* Urbana-Champaign, IL, May 2005.

[3] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-efficient communication protocol for wireless sensor networks. In *Proceedings of IEEE Hawaii International Conference System Sciences,* Hawaii, January 2000.

[4] W. Heinzelman, J. Kulik and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th ACM/IEEE Annual International Conference on Mobile Computing and Networking(MobiCom'99),* Seattle, WA, August 1999.

[5] C. Intanagonwiwat, R. Govindan and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking(MobiCom'00),* August 2000, Boston, Massachusetts.

[6] Y. Kotidis. Snapshot Queries: Towards Data-Centric Sensor Networks. In *Proceedings of the 21th IEEE International Conference on Data Engineering(ICDE),* 2005.

[7] H. Luo, F. Ye, J. Cheng, S. Lu and L. Zhang. TTDD: Two-Tier Data Dissemination in Large-Scale Wireless Sensor Networks. In *Proceedings of ACM International Coference on Mobile Computing and Networking (MobiCom'02),* Atlanta, Georgia, USA, September 2002.

[8] A. Manjeshwar and D. P. Agrawal. TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks. In *Proceedings of the 1st IEEE International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing,* San Francisco, CA, April 2001.

[9] A. Manjeshwar and D. P. Agrawal. "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks,. In *Proceedings of the 2nd IEEE International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing,* Ft. Lauderdale, FL, April 2002.

[10] A. Meka and A. K. Singh. Distributed Spatial Clustering in Sensor Networks. In *Proceedings of the 10th International Conference on Extending Database Technology(EDBT),* 2006.

[11] O. Younis and S.Fahmy. HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-hoc Sensor Networks. *IEEE Transactions on Mobile Computing,* vol. 3, no. 4, pp. 366-379, Oct-Dec 2004.

[12] S. Yoon and C. Shahabi. Exploiting Spatial Correlation Towards an Energy Efficient Clustered AGgregation Technique(CAG). In *Proceedings of IEEE International Conference on Communications(ICC),* 2005.

[13] JSIM: A Java-based simulation and animation environment. *http://www.cs.uga.edu/ jam/jsim/.*