國立交通大學

網路工程研究所

碩 士 論 文

在 無 線 感 測 網 路 下 建 立
最 大 生 存 時 間 之 資 料 蒐 集 多 叢 樹

Constructing Localized Maximum-Lifetime Data Aggregation Forests in Wireless Sensor Networks

研 究 生：劉嵩裕

指導教授：黃俊龍　教授

中 華 民 國　一百　年 九 月

在無線感測網路下建立最大生存時間之資料蒐集多叢樹
Constructing Localized Maximum-Lifetime Data Aggregation Forests in Wireless Sensor Networks

研 究 生：劉嵩裕　　Student：Song-Yu Liu

指導教授：黃俊龍　　Advisor：Jiun-Long Huang

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年九月

在無線感測網路下建立最大生存時間之資料蒐集多叢樹

學生：劉嵩裕　　指導教授：黃俊龍 博士

國立交通大學網路工程研究所

摘要

在無線感測網路下，電源使用效率是個很重要的議題， 特別是在環境監控之類的應用下。 在這種環境下，資料蒐集是最基本的操作，每個感測器會感測環境， 並定期將資料回報給基地台。 樹狀結構的資料蒐集方法常常用來節省電力消耗，以達到延長感測器生存時間。 每個節點會選定一個母節點，並將資料傳送給母節點； 母節點負責從子節點蒐集資料並匯整，以減少不必要的傳輸。

在這篇論文中，我們研究如何在多個基地台下的無線感測網路中建立資料蒐集多叢樹。我們的目標是延長感測網路的生存時間，生存時間定義爲感測網路起始後至第一個節點耗盡電力的期間。 這個問題已被證明爲NP-complete，因此我們提出了分散式的演算法來有效率地建立資料蒐集多叢樹。爲了延長感測網路生存時間，電量較多的節點應該負責支持較多的子節點。 基於這樣的概念，每個節點透過和鄰近節點交換訊息，藉此獲得局部資訊，再經過訊息交換後，每個節點可以透過鄰近節點表來決定母節點。 此外，我們的方法考慮到節點壞掉或者加入的情況；也針對電量快耗盡的節點進行調整，以達到延長生存時間的效果。 最後我們透過數個實驗來驗證方法，實驗結果也顯示我們的方法能有效率地建立資料蒐集多叢樹，並延長感測網路的生存時間。

關鍵字：網路生存時間，無線感測網路，分散式演算法

Constructing Localized Maximum-Lifetime Data Aggregation Forests in Wireless Sensor Networks

Student: Song-Yu Liu Advisor: Dr.Jiun-Long Huang

Institutes of Network Engineering
National Chao Tung University

ABSTRACT

In wireless sensor networks, energy efficiency is a critical issue, especially for applications such as environmental monitoring. Data gathering is the fundamental operation in these applications. It means that sensor nodes sense phenomena, and periodically report the sensed data to the base stations. In order to conserve energy max maximize sensor lifetime, a tree-based data aggregation is often used for collecting data from sensor nodes. Every node determines a parent node to transmit data while a parent node aggregates the data from child nodes to eliminate redundant transmission. In our work, we study the construction of a data aggregation forest where there are several base stations in a sensor network. We aim to prolong the network lifetime of a sensor network while the definition of network lifetime is the time until the first node depletes its energy. The problem is proven to be NP-complete. Hence, we propose a distributed algorithm which constructs a data aggregation forest efficiently. To increase the network lifetime, the nodes with more energy should also keep more child nodes. Based on the concept, each node exchanges messages between its neighbor nodes to obtain local information. After the exchange, every node decides a parent node according to its neighbor table. Our solution includes procedure of route maintenance while there is node failure or addition. Moreover, forest adjustment is also adapted to consider the low energy nodes for prolonging the network lifetime. Finally, we evaluate our solution with numerical simulations. The results show that our algorithm can efficiently construct a data aggregation forest which prolongs the network lifetime.

Keywords: network lifetime, wireless sensor network, distributed algorithm

誌謝

很高興就讀交通大學，能在這樣優質的研究學術環境下學習，也要感謝黃俊龍教授兩年來的指導，如果沒有黃俊龍教授，就沒有今天的我。除了研究方面，眞的要感謝黃俊龍教授給我機會參與計畫，讓我有更多的磨練，有這樣亦師亦友的指導教授，眞的很棒！另外也要感謝我的口試委員：曾建超教授以及胡誌麟教授，在口試的時候提供許多寶貴的意見，讓我可以更加精進。

還要感謝實驗室的各位，振哲學長、孝文學長，有學長的指導，解決了我在學業上的種種疑惑；棕樺、傻西，讓我在研究路上不孤單，大家總是能一起討論研究上的課題；還有各位學弟們，謝謝大家的支持和幫忙！

最後我要感謝我的家人和朋友，你們的支持和鼓勵，給了我許多幫助，有了你們，我才能順利完成這份論文，謝謝！

Contents

摘要 i

Abstract ii

誌謝 iii

Table of contents iv

List of figures vi

1 Introduction 1

2 Related Work 4

3 System Model And Problem Statement 7

3.1 System Model 7

3.2 Problem Statement 8

4 Distributed Algorithm For Forest Construction 10

4.1 Overview 10

4.2 Initialization 12

4.3 Forest Construction 14

4.4 Refinement 16

4.5 Route Maintenance 20

4.5.1 Node Failure 20

4.5.2 Node Addition 21

4.5.3 Forest Adjustment 24

5 Simulation Results **27**

5.1 Impact of Number of Sensor Nodes 28

5.2 Impact of Number of Base Stations 29

5.3 Impact of Transmission Range 30

5.4 Impact of Network Area Size 32

6 Conclusion **34**

List of Figures

4.1 Level structure of sensor nodes 12
4.2 Illustration of Algorithm 2 16
4.3 Illustration of Algorithm 3 19
4.4 Illustration of Algorithm 4 for node failure 21
4.5 Illustration of Algorithm 5 for node addition 23
4.6 Illustration of Algorithm 6 for forest adjustment 25

5.1 Simulation with difference number of nodes 28
5.2 Simulation with difference number of base stations 29
5.3 Simulation with difference transmission range 30
5.4 Simulation with difference network area size 32

Chapter 1

Introduction

Wireless sensor networks can be applied in different environments while the applications can be various, such as forest fire detection, home automation, and factory process control [1]. In a wireless sensor network, there are a large number of sensor nodes which can sense nearby environment and communicate with another nodes. In addition, there are base stations to control the sensor network and process data from every node. One of the common and most important applications is environmental monitoring, which employs numerous sensor nodes for continuous sensing.

Data gathering is the fundamental operation in such applications. It means that the sensor nodes sense phenomena (e.g., temperature, humidity) and report the sensed data to a base station at regular intervals for advanced processing. However, for such applications, the sensor nodes usually equip batteries with limited energy, and it might not be possible to replace or recharge the batteries. To prolong sensors lifetime, the energy consumption becomes a critical issue [2]. It is not efficient to transmit the sensed data to the base stations for all the sensor nodes. Since the data generated from neighboring nodes may be the same or highly correlated, sending each packet individually results in redundant overhead. In order to conserve energy and maximize sensor lifetime, data aggregation is used for minimizing the number of communications. Data aggregation is fusing the data from several sensor nodes to eliminate redundant transmission. That is, the fewer transmissions imply the fewer energy consumption. In this paper, we adopt perfect aggregation while gathering data. It means that every sensor node aggregates the received data into one single packet with the same size.

There are several studies of energy efficient data aggregation: *cluster-based* [3], *chain-*

based [8], and *tree-based* [10]. In data aggregation, the tree-based topology often performs well in terms of network lifetime [10, 11]. After sensor nodes deployment, the tree is constructed while taking the base station as the root. There might be multiple base stations in sensor network applications [14]. In the data aggregation forest, each sensor node joins a tree rooted at a base station while each tree does not intersect with each other. Hence, the problem we study here is the data aggregation forest construction which maximizes the network lifetime. The definition of network lifetime can be various [11]. One is defined as the time until the first node depletes all of its energy. Another definition is that the time until the network is partitioned where some sensor nodes cannot report sensing data to the base station. The former definition is adopted in this paper.

This problem is proven to be NP-complete [13] while the computation cost of solving NP-complete problem is too expensive. In the forest construction, there are several issues must be considered. First, the algorithm must be scalable. When increasing number of nodes, the message transmission and computation should also be scale. The centralized algorithm could result in high cost while the sensor network is in large scale. The requirement of global knowledge might lead to a large amount of message overhead. Another issue is robustness, which means that the algorithm should be resilient to node addition or failure. Moreover, if there are some sensor nodes join or leave, it needs to execute the entire computation again while centralized algorithm is adopted. As a consequence, our algorithm proposed in this paper constructs a forest topology which prolongs network lifetime. The algorithm is distributed since centralized manner is too costly in a dynamic network. In addition, the framework can handle node addition/failure efficiently.

Our work can be divided into two parts: *topology construction* and *route maintenance*. In topology construction, the goal is to construct disjointed data-aggregation trees rooted at each base station. There are three phases in this part: *1) Initialization*. Via broadcasting messages, each base station informs every node to construct its one-hop neighbor table. The neighbor table is used for determining a parent node in next phase. *2) Construction*. After initialization, each node selects a parent node base on its neighbor table. *3) Refinement*. Since there could be some nodes with low remaining energy and high degree, we make the refinement on these nodes to balance the local structure.

Further, the algorithm of route maintenance is also proposed: *1) Node failure*. When there is node failure, its neighboring nodes can detect the status by exchanging messages. Then, the neighboring nodes can enable the nodes, which selects the failed node as the

parent, to reconstruct the local topology. *2) Node addition.* The new node obtains its local information by broadcasting messages. The same as initialization, the new node informs its neighboring nodes to update their neighbor table. After that, the neighboring nodes reconstructs the local structure. *3) Forest adjustment.* Since our goal is to prolong the network lifetime, adjusting the loading of a low energy node is necessary. When the energy level of a node is lower than a threshold, the child nodes select a new parent to reduce its degree.

The rest of this paper is organized as follow. Section 2 presents the related work of this study. Section 3 describes the system model and problem definition of forest construction. The proposed algorithm of forest construction is presented in Section 4. Simulation results are given in Section 5. Finally, Section 6 makes a conclusion for this paper.

Chapter 2

Related Work

Efficient data aggregation in a sensor network has been extensively studied. Rajagopalan and Varshney [9] describe the three different data aggregation protocols: *cluster-based*, *chain-based*, and *tree-based*. Heinzelman et al. [3] presented a cluster-based protocol called Low-Energy Adaptive Clustering Hierarchy (LEACH). LEACH protocol is distributed and organizes sensor nodes into clusters. Each node randomly decides to become a cluster head which has responsibility to transmit aggregated data to the base station. The cluster heads broadcast messages to all the other sensors. Then, noncluster head nodes join the nearest cluster by the signal strength of the advertisement received, and send the data to the cluster head. LEACH protocol assumes each sensor node is capable of being a cluster head. However, in the energy-constrained sensor networks, the assumption might not be valid. Since the randomized cluster head rotation, LEACH protocol is far from being optimal.

In chain-based protocol, each sensor node transmits data only to its nearest neighbor. Lindsey and Raghavendra [8] proposed a chain-based data aggregation protocol called Power-Efficient Data-Gathering Protocol for Sensor Information Systems (PEGASIS). In PEGASIS, sensor nodes are organized into a linear chain in a centralized manner. The farthest node from the base station starts to send the data. At each data-gathering round, sensor nodes aggregate received data and transmit to its neighbor along the chain. The nearest node on the chain finally transmits the aggregated data to the base station. Although PEGASIS has great energy savings, it has to collect the global information about all node positions. Moreover, to minimize the maximum neighbor distance on the chain is NP-complete (traveling salesman problem), and the latency might also be a

difficulty.

Tree-based data aggregation is often adopted where tree-based protocols do no need to maintain a routing table at each node. Tan and Korpeoglu [10] addressed a power-efficient data gathering and aggregation protocol (PEDAP). PEDAP is a minimum spanning tree-based protocol where the goal is to maximize the network lifetime. The remaining energy of each node should be considered while data gathering. Therefore, a power-aware version of PEDAP is also proposed, which is called PEDAP-PA. In order to balance the load, PEDAP-PA re-computes the routing tree after a period of time. However, the base station should know the location of all nodes in PEDAP. More related studies of tree-based data aggregation like an aggregation scheduling based on tree topology addressed by Li et al. [6]. The proposed aggregation scheduling considers the wireless sensor networks with physical interference model. Zhang et al. [15] proposed a distributed progressive algorithm. The goal is different from maximizing network lifetime, but to maximize the lifetime vector which is defined as the lifetimes of all sensors, sorted in ascending order. The distributed algorithm introduced in [4] constructs an approximate minimum spanning tree which can be used in data aggregation protocol. Liang et al. [7] studied the problem of constructing a maximum lifetime tree for data gathering but without aggregation. In that scenario, data reduction is not allowed, e.g., gathering images from monitored region.

Wu et al. [13] studied the construction of a data gathering tree which maximizes the network lifetime. The problem is proved as NP-complete in that work. Further, the authors design an algorithm which starts from an arbitrary tree. Since energy consumption of each node is in direct proportional to the number of child nodes, the algorithm reduces the number of high degree nodes iteratively. Tan et al. [11] presented a distributed energy-efficient routing approach called Localized Power Efficient Data Aggregation Protocol (L-PEDAP). The routing scheme takes advantage of localized topology such as relative neighborhood graph (RNG) [12] and local minimum spanning tree (LMST) [5]. L-PEDAP can approximate minimum spanning tree and can be computed efficiently by neighbor information. Based on localized topology RNG or LMST, the authors provide several parent selection strategies to construct a routing tree. In addition, L-PEDAP also considers the remaining energy of each node. However, the studies mentioned above only consider a sensor network with one base station. Wu et al. [14] extended the tree construction to the forest construction which maximizes the network lifetime of a wireless sensor network. The authors consider the case where there are multiple base stations.

Through broadcasting beacons, each base station constructs its local tree where each tree does not intersect with another trees. After gathering the information of each node and its neighbors, the algorithm finds out the bottleneck nodes which might consume energy rapidly since low residual energy or high degree. Then, the algorithm makes improvement on the bottleneck nodes to prolong network lifetime. The construction algorithm is centralized and near optimal which is proved in that work. In contrast to these approaches, we focus on a sensor network with multiple base stations. Furthermore, we propose a distributed algorithm to construct data aggregation forest efficiently.

Chapter 3

System Model And Problem Statement

3.1 System Model

The following are the assumptions about the system. Consider a sensor network with N sensor nodes and several base stations. The base stations and sensor nodes are stationary. The base stations have unlimited energy while each node i has limited energy $E(i)$. We assume the values $E(i)$ are the same since there are homogeneous sensor nodes. Each node periodically senses the nearby environment and reports the data to the base station in each *round*. A *round* is a period of time while a node can receive data from child nodes and send the aggregated data to the parent node. In a single *round*, the energy consumption of a node can be divided into two parts:

1) **Receive**: In the monitoring, a node receives one B-bit data from each child node. Assume $C(i)$ is the number of child nodes of node i, and r is the energy required to receive one bit of data. The energy consumption of receiving of node i in a *round* is $rBC(i)$.

2) **Send**: After receiving data from child nodes, each node aggregates the received data, and send the aggregated data to the parent node. We adopt a perfect aggregation model which means a node combines $C(i)$ incoming data, each data size is B bits, with its data into one message of size B bits. The practical models can be min,

max, or sum. Assume t is the energy required to transmit one bit of data, and every node has only one parent node. Hence, the energy consumption of sending of node i in a *round* is tB .

In [1], energy consumption of a sensor node can be divided into three parts: *sensing*, *communication*, and *data processing*. A sensor node depletes the maximum energy in communication which involves data transmission and reception. Hence, we only consider the energy consumption of communication in our system. The total energy consumption for node i in a single *round* is $rBC(i) + tB$. Given energy value $E(i)$, the total round of node i before energy depletion is computed as $\frac{E(i)}{rBC(i)+tB}$.

There are various definitions of network life time in the context of sensor networks. One of the definitions is the time until the first node depletes its energy. Another definition is the time until the network is partitioned while there are some nodes cannot report their data to the base station. We adopt the former definition in our system.

N	number of nodes
$E(i)$	remaining energy of node i
$C(i)$	number of child nodes of node i
r	energy required to receive one bit of data
t	energy required to transmit one bit of data
B	message size in bit

Table 3.1: List of symbols

3.2 Problem Statement

We consider a sensor network with N nodes. Each sensor node periodically senses the environment, and generates data in each round. To collect data from the sensor nodes, we construct a data-aggregation forest which is a tree-based topology. A data-aggregation forest consists of several data-aggregation trees rooted as the base stations. Each data-aggregation tree does not intersect with another trees. It implicitly specifies that each node joins only one data-aggregation tree. In a single tree, each node receives the data from its child nodes, combines them with its data, and sends the aggregated message to the parent node on its way to the base station. The practical applications are event detection systems or environmental monitoring systems. In these applications, we need

to keep the coverage of sensor nodes which means prolonging the alive time of each node as long as possible.

The problem is to find a forest topology which maximizes the network lifetime while the problem is proven to be NP-complete in [13]. The network lifetime is defined as the time until the first node depletes its energy while $L(i) = \frac{E(i)}{rBC(i)+tB}$ is the total round of node i. In order to maximize the network lifetime, we have to construct a data-aggregation forest which maximizes the minimum $\frac{E(i)}{rBC(i)+tB}$. From the definition of the total round, the more energy $E(i)$ a node keeps, the more child nodes $C(i)$ it should have.

Chapter 4

Distributed Algorithm For Forest Construction

4.1 Overview

Our aim is to construct a data-aggregation forest in a distributed manner. The proposed approach consists of two parts: *topology construction* and *route maintenance*. The goal of topology construction is to construct disjointed data-aggregation trees rooted at each base station, which means each node determines a parent node. There are three phases in this part:

1) **Initialization**: At the initial stage, all nodes do not know their surroundings. After broadcasting *HELLO* messages by the base stations, each node can obtain its one-hop neighbor table. According to the neighbor tables, all nodes are informed of their hop levels, potential parent nodes, and potential child nodes.

2) **Construction**: In this phase, the base stations broadcast *INVITE* messages to enable forest construction. After a period of time *INVITE TIME THRESHOLD*, the nodes can hear all messages from potential parent nodes. Each node then selects a parent node by these *INVITE* messages, and send a *REPLY* message back to the parent node. Upon receiving *REPLY* messages, the parent nodes record its actual number of child nodes.

3) **Refinement**: After construction, there could be some nodes with imbalance number of child nodes and remaining energy between neighbor nodes. This phase refines the imbalance tree structure. All nodes update their neighbor table with actual number of child nodes from *UPDATE* messages broadcasted by the base stations. The nodes with imbalance structure broadcast *REFINE* messages to inform their child nodes to select a parent again. If child nodes select new parent nodes, child nodes broadcast *CHANGE* messages to update the actual topology.

After topology construction, there could be node failure, node addition, or node with low energy. The topology may be changed to adapt to these conditions. Hence, route maintenance is necessary to deal with the three issues efficiently:

1) **Node failure**: A node may disappear due to energy depletion or damaged by external factors. Therefore, the nodes should detect node failure actively by exchange *KEEP-ALIVE* messages periodically. If there is a node disappears, the neighbor nodes broadcast *INVITE* messages to reconstruct the topology.

2) **Node addition**: There could be a new node deployment. In our algorithm, it doest not need to reconstruct whole topology but construct the topology below to the new node only. The new node broadcasts an *ASK* message to obtain its hop level. Through broadcasting *HELLO* message, neighbor nodes of new node update their neighbor tables. The potential parent nodes of new node then broadcast *INVITE* messages to invoke new node to select a parent. After parent selection of new node, it broadcasts *REFINE* message to reconstruct the topology below itself.

3) **Forest adjustment**: To prolong the network lifetime, adjusting the loading of a low energy node is necessary. In this phase, we want to reduce the degree of the low energy node to prolong its lifetime. When the energy level of a node is lower than a threshold, it broadcasts a *LEAVE* message to notify the neighbor nodes of its low energy. The neighbor nodes located at the same level then broadcast *ADJUST* messages with their remaining energy and number of child nodes. The child nodes of the low energy node waits for a period of time to ensure that the *ADJUST* messages are received. After that, the child nodes can select a new parent to reduce the degree of the low energy node.

4.2 Initialization

In this phase, each node exchanges information between each other to obtain its *potential parent nodes*, *potential child nodes*, and *hop level*. In the beginning of the initialization, each base station broadcasts *HELLO* messages to inform all nodes of their neighbor environment. A *HELLO* message includes three fields: a *base station ID*, a *sender ID*, and the *hop count* to the base station. A *base station ID* specifies which base station broadcasts the *HELLO* message. In the flooding of *HELLO* messages, a *sender ID* indicates the source of the *HELLO* message. The *hop count* is used for keeping track of which level the node is located at.

After each node receives *HELLO* messages from its neighbor nodes, it records the minimum hop count to arbitrary base station as its hop level. Then, each node broadcasts a *HELLO* message with its ID and level. All node maintain their one-hop neighbor table. Each entry includes a *ID* and the corresponding *hop level*. According to the level structure and one-hop neighbor table, each node knows the number of potential child nodes of itself.

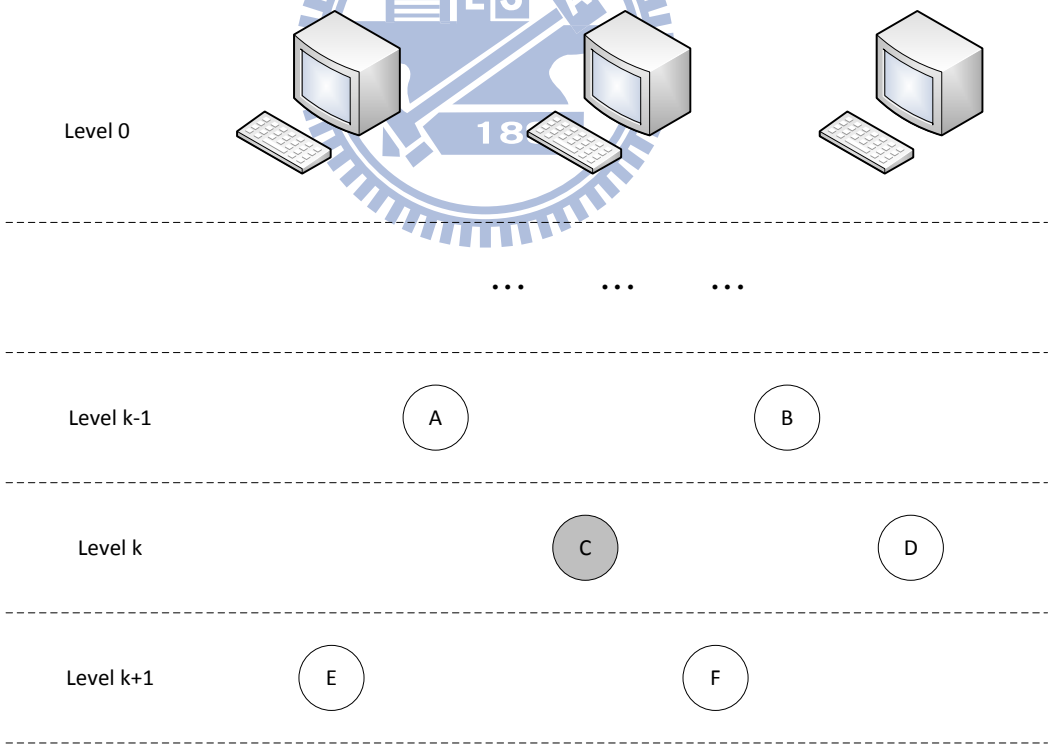

Figure 4.1: Level structure of sensor nodes

ID	Level
A	k-1
B	k-1
D	k
E	k+1
F	k+1

Table 4.1: Neighbor table of node C after initialization

Figure 4.1 illustrates the level structure of sensor nodes. The base stations are located at level 0. A node located at level 1 means the hop count to the nearest base station is 1. The level increases while the hop count to the base station adds. For example in Figure 4.1, nodes A and B are located at level k-1. Nodes C and D are located at level k. Nodes E and F are located at level k+1. For node C, it maintains an one-hop neighbor table after initialization as shown in Table 4.1. The nodes at level higher than level of node C by 1 are called *potential parent nodes*. In Figure 4.1, that are nodes A and B. The nodes at level lower than level of node C by 1 are called *potential child nodes*. In Figure 4.1, that are nodes E and F. Node D, located at level k which is the same as node C located at, is called *neighbor nodes*. The intention of the level structure is to avoid the cycle in the forest. The nodes only choose one node at higher level as the parent. The nodes do not choose another nodes at the same or lower level as the parent since it increases the depth of the forest. The level structure limits the depth of the forest. It implicitly reduces the latency of the reporting time from leaf nodes to the base stations.

Algorithm 1 : Forest Construction - Initialization

Input:
G : connected network
BS : set of all base stations
N : set of sensor nodes
Output:
Each node $\in N$ maintains an one-hop neighbor table

```
1: for each base station b, b ∈ BS do
2:     Broadcast HELLO message
3: end for
4: for each sensor node n, n ∈ N do
5:     if HELLO message H received then
6:         Record the minimum hop count to arbitrary base station as its hop level
7:         Insert SenderID(H) and HopLevel(H) into its neighbor table
8:         Record the number of potential child nodes
9:         Broadcast HELLO message with its ID and level
10:    end if
11: end for
```

4.3 Forest Construction

The main goal in this phase is to construct the forest topology which means each node determines a parent node. At the end of this phase, the data aggregation trees rooted at each base station are constructed. Each base station broadcasts *INVITE* messages to invoke the procedure of parent selection. An *INVITE* message includes a *base station ID*, a *sender ID*, the *number of potential child nodes*, and the *remaining energy*. A base station ID specifies which base station broadcasts the *INVITE* message. A sender ID records which node forwards the *INVITE* message. The last two fields in *INVITE* message are the criteria to select a parent.

When each node receives *INVITE* messages, it updates the number of potential child nodes of corresponding entry in the neighbor table, and broadcasts an *INVITE* message with its ID, the number of potential child nodes, and the remaining energy. After a period of time called *INVITE TIME THRESHOLD*, all nodes select the parent nodes according to the parent select function. Since data-aggregation trees are disjointed which means a node can only join one tree, each node just determines one parent node. In the parent selection, each node decides a parent node with a probabilistic method.

To maximize the network lifetime, a node should choose a parent with more energy

and less potential child nodes. In Section 3.1, we give the definition of total rounds of node i: $\frac{E(i)}{rBC(i)+tB}$. E(i) is the remaining energy of node i. C(i) is the number of children for node i. r is the energy required to receive one bit of data. t is the energy required to send one bit of data. B is the message size in byte. In the forest construction, each node does not know how many child nodes it will possess. Hence, we can substitute C(i) for the number of potential child nodes PC(i). This means the potential round of a node is $\frac{E(i)}{rBPC(i)+tB}$. The node with higher potential round should have more child node. Therefore, when a node decides a parent, it selects a node as the parent in direct proportion to $\frac{E(i)}{rBPC(i)+tB}$. After parent selection, each node sends a *REPLY* message with sender and receiver ID back to the parent node. The node receiving a *REPLY* message accumulates the actual number of child nodes. At this step, the data aggregation forest is constructed. It means that each node will receive data from its child nodes, and know to which node it sends the aggregated data.

Algorithm 2 : Forest Construction - Construction

Input:
G : connected network
BS : set of all base stations
N : set of sensor nodes
Output:
Each node $\in N$ selects a parent node

1: **for** each base station b, $b \in BS$ **do**
2: Broadcast *INVITE* message
3: **end for**
4: **for** each sensor node n, $n \in N$ **do**
5: **while** *INVITE* message I received in *INVITE TIME THRESHOLD* **do**
6: Update NumberOfPotentialChildNode(I) in its neighbor table
7: Broadcast *INVITE* message with its ID, the PC, and the remaining energy
8: **end while**
9: Select a parent node
10: Send *REPLY* message back to the parent node
11: **if** *REPLY* message R received **then**
12: Record the actual number of child nodes
13: **end if**
14: **end for**

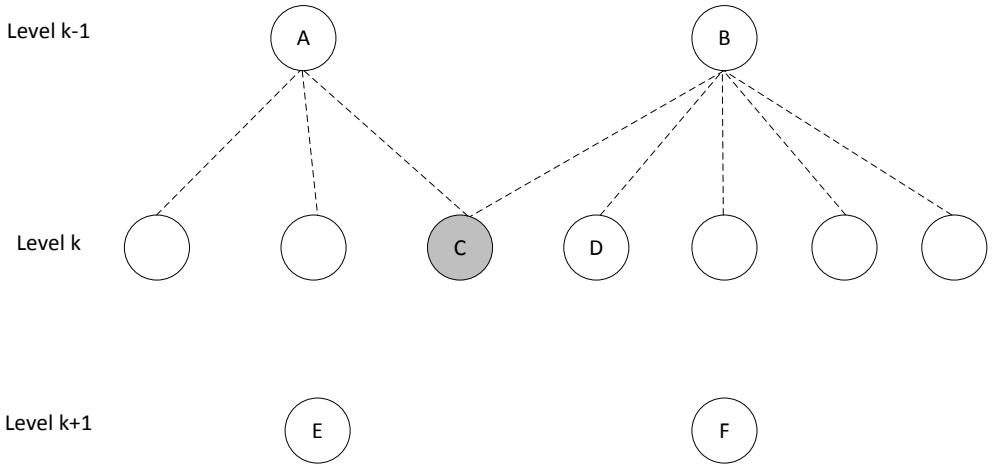

Figure 4.2: Illustration of Algorithm 2

ID	Level	Number of potential child nodes
A	k-1	3
B	k-1	5
D	k	2
E	k+1	2
F	k+1	1

Table 4.2: Neighbor table with PC of node C in Figure 4.2

Table 4.2 shows the neighbor table of node C which is located at level k. In a period of time *INVITE TIME THRESHOLD*, node C updates all PC entry of its neighbor table. Since node C is located at level k, there are two potential parent: A and B.

The potential round of node A is $\frac{E(A)}{rBPC(A)+tB} = X$.

The potential round of node B is $\frac{E(B)}{rBPC(B)+tB} = Y$.

Node C selects a parent between A and B in direct proportion to X and Y. The probability to select node A as the parent is $\frac{X}{X+Y}$. The probability to select node B as the parent is $\frac{Y}{X+Y}$.

4.4 Refinement

After the construction of data-aggregation forest, there could exist some nodes with imbalance number of child nodes and the remaining energy between its neighbor nodes. In

this phase, the goal is to adjust the imbalance topology. Each base station broadcasts *UPDATE* messages to enable all nodes to update their actual neighbor structure. A *UPDATE* message includes a *sender ID*, the *actual number of child nodes*, the *remaining energy*. A *sender ID* specifies which nodes sends the *UPDATE* message. The *actual number of child nodes* is used for updating the actual topology in the one-hop neighbor table. The *remaining energy* enable the accurate round calculation.

While each node receives *UPDATE* messages, it updates the actual number of child nodes in the neighbor table, and broadcasts a *UPDATE* message with its ID, the actual number of child nodes, and the remaining energy. In a period of time called *REFINE TIME THRESHOLD*, each node finds the maximum round (max round) at the same level in its neighbor table. If the difference between its round and max round greater than or equal to *ROUND DIFFERENCE THRESHOLD*, the node broadcasts a *REFINE* message to enable topology refinement. When the node receives a *REFINE* message, the node selects the parent again by the parent select function while the refinement may occur in the same tree or different tree. It means that a node may join another tree after the refinement. If the node selects a new parent, it broadcasts a *CHANGE* message to inform former parent and new parent. Therefore, former parent and new parent can update their actual number of child nodes.

Algorithm 3 : Forest Construction - Refinement

Input:

G : connected network

BS : set of all base stations

N : set of sensor nodes

Output:

Each node $\in N$ balances the number of child nodes between its neighbor nodes

1: **for** each base station b, $b \in BS$ **do**
2: Broadcast *UPDATE* message
3: **end for**
4: **for** each sensor node n, $n \in N$ **do**
5: **while** *UPDATE* message U received in *REFINE TIME THRESHOLD* **do**
6: Update the actual number of child nodes in its neighbor table
7: Broadcast *UPDATE* message with its ID and the actual number of child nodes
8: **end while**
9: Find the maximum round(max_round) at the same level in its neighbor table
10: **if** $|its\_round - max\_round| \geq$ *ROUND DIFFERENCE THRESHOLD* **then**
11: Broadcast *REFINE* message
12: **end if**
13: **if** *REFINE* message R received **then**
14: Select a parent
15: **if** A new parent is selected **then**
16: Broadcast *CHANGE* message
17: **end if**
18: **end if**
19: **end for**

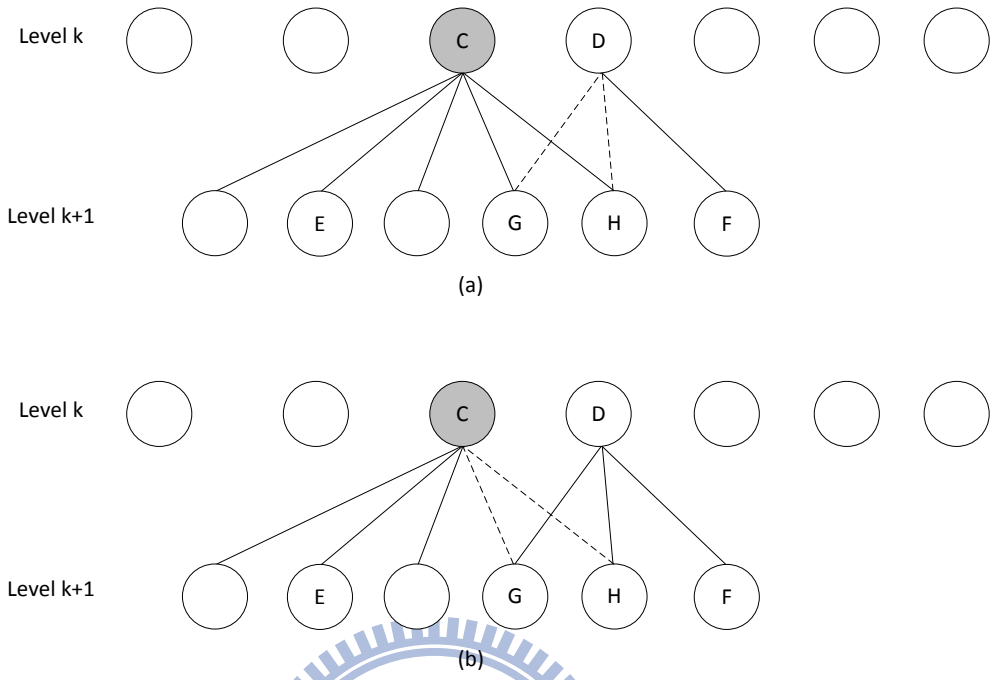

Figure 4.3: Illustration of Algorithm 3

We give an example to illustrate the refinement. Figure 4.3(a) shows a constructed topology. There are 5 nodes select node C as the parent, and there is 1 node selects node D as the parent. When the base stations broadcast *UPDATE* messages, nodes C and D receive *UPDATE* messages in *REFINE TIME THRESHOLD*. After the period of time *REFINE TIME THRESHOLD*, node C finds the minimum round at the same level in its neighbor table. The round of node C is $\frac{E(C)}{rBC(C)+tB} = X$. The maximum round is from node D: $\frac{E(D)}{rBC(D)+tB} = Y$. Node C subtracts its round from the maximum round $X - Y$. The difference $X - Y$ is less than *ROUND DIFFERENCE THRESHOLD*. Therefore, node C broadcasts a *REFINE* message. Nodes G and H receive *REFINE* messages and select node D as the new parent. In Figure 4.3(b), node G and H select node D as the parent. The original ratio of C(C) to C(D) is 5:1. After the refinement, the ratio of C(C) to C(D) is 1:1.

4.5 Route Maintenance

4.5.1 Node Failure

After setting up the data-aggregation forest, there could be node failure or node addition. The algorithm maintains the forest topology, even deleting or adding a node to the sensor network. To detect node failure actively, each node periodically broadcasts *KEEP-ALIVE* messages. If a node doest not receive *KEEP-ALIVE* messages from the neighbor node in a period of time, the neighbor node is taken as a disappeared node. The nodes, which are located at the same level as the disappeared node located at, could be the potential parent to those nodes select the disappeared node as the parent. Hence, these nodes broadcasts *INVITE* messages with the actual number of child nodes. Then, the nodes lost their parents receives *INVITE* messages. These nodes can update their neighbor tables, and select a new parent.

Algorithm 4 : Route Maintenance - Node Failure

Input:
G : connected network
N : set of sensor nodes
Output:
Nodes in N select the fail node as the parent construct their local topology

```
1: for each sensor node n, n ∈ N do
2:     Broadcast KEEP-ALIVE message periodically
3:     if a node X disappears then
4:         Node at the same level broadcasts INVITE messages with the actual number
   of child nodes
5:         if INVITE message I received then
6:             Select a parent according to new neighbor table
7:         end if
8:     end if
9: end for
```

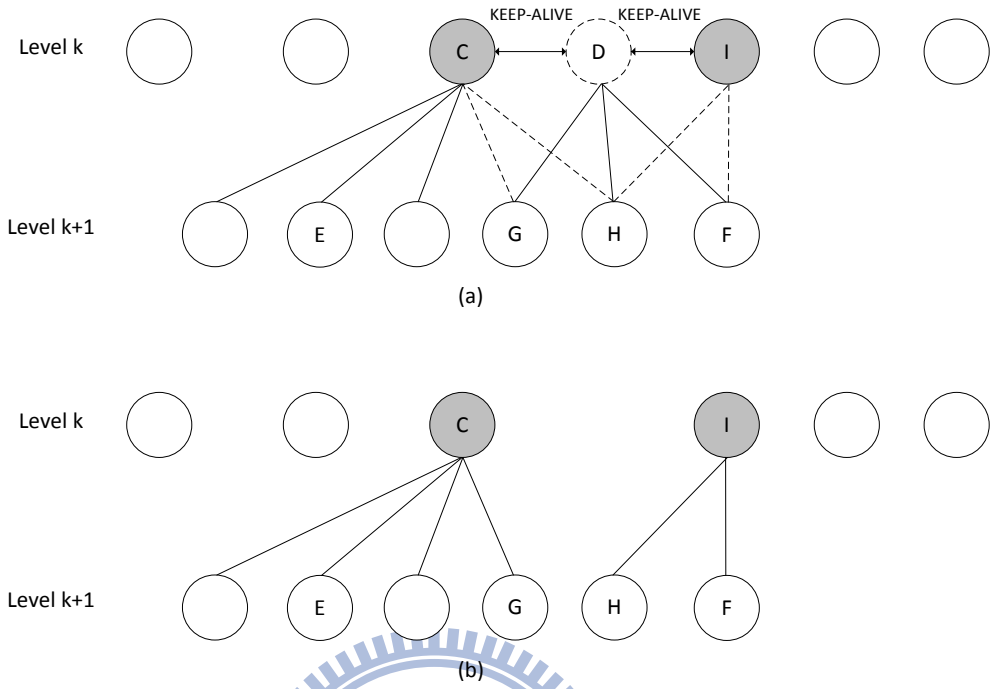

Figure 4.4: Illustration of Algorithm 4 for node failure

Figure 4.4 illustrates the route maintenance when there is a node fail. In Figure 4.4(a), node D broadcasts *KEEP-ALIVE* messages between its neighbor nodes. When node D disappears, nodes C and I broadcast *INVITE* messages. While receiving *INVITE* messages, nodes G, H, and F select new parents as shown in Figure 4.4(b).

4.5.2 Node Addition

Consider the case of node additions. If a new sensor node is deployed, it broadcasts an *ASK* message with its ID to obtain which level it is located at. The node receiving an *ASK* message send an *ANSWER* message with its base station ID and level back to the new node. While receiving *ANSWER* messages, new node records the minimum hop level and plus 1 as its level. After that, new node broadcasts a *HELLO* message. The neighbor nodes of new node will receive *HELLO* messages, and update their neighbor tables. The nodes, whose hop level higher than new node by 1, are the potential parent of the new node. These nodes broadcast *INVITE* messages to inform new node to select a parent. New node can choose parent according to *INVITE* messages, and broadcasts a *INVITE*

message to reconstruct the topology below the new node.

Algorithm 5 : Route Maintenance - Node Addition

Input:
G : connected network
N : set of sensor nodes
Output:
Node added to N constructs its local topology

```
1: for each sensor node n, n ∈ N do
2:     if a node X adds then
3:         X broadcasts ASK message
4:         if ASK message received then
5:             Send ANSWER message back to X
6:         end if
7:         if ANSWER message received then
8:             X record the minimum hop level and plus 1 as its level
9:         end if
10:        X broadcasts HELLO message
11:        if HELLO message I received then
12:            Update its neighbor table
13:        end if
14:        for node with level higher than X by 1 do
15:            Broadcast INVITE message
16:            if INVITE message I received then
17:                X selects a parent
18:                X broadcasts REFINE message
19:                Nodes below to X reconstruct the topology
20:            end if
21:        end for
22:    end if
23: end for
```

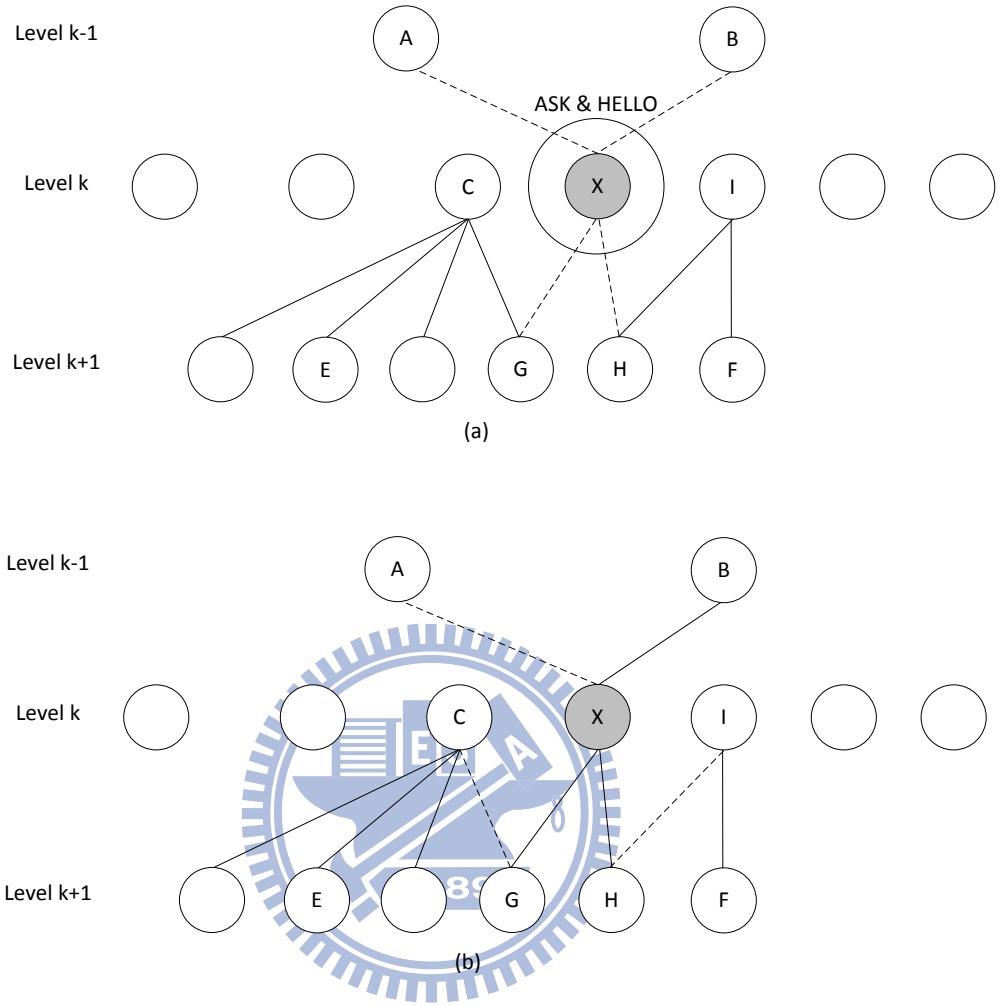

Figure 4.5: Illustration of Algorithm 5 for node addition

In Figure 4.5(a), we add node X into the sensor network. Node X broadcasts an *ASK* message to obtain its level. After that, node X broadcasts a *HELLO* message to enable neighbor nodes update their neighbor tables. Nodes A and B are located at level k-1 which is higher than the level node X located at by 1. Therefore, nodes A and B broadcast *INVITE* messages to inform node X to select a parent. In Figure 4.5(b), node X selects node B as the parent. Then, node X broadcasts a *REFINE* message, and nodes below node X reconstruct their topology. As shown in Figure4.5(b), nodes G and H select node X as their parent.

4.5.3 Forest Adjustment

The low energy nodes is the bottleneck while prolonging the network lifetime. Hence, we need to reduce the degree of a low energy node. Through forest adjustment, the child nodes of a low energy node may select a new parent. Therefore, the lower degree of a low energy node implies the longer lifetime. Once a node with energy level lower than a threshold, the node broadcasts a *LEAVE* message. The neighbor nodes, located at the level equals to that of the lower energy node plus one, remove the entry of the low energy node from their neighbor tables. The neighbor nodes with the same level, which receives *LEAVE* messages, broadcast *ADJUST* message. An *ADJUST* message includes a sender ID, the remaining energy, and the number of child nodes. The three fields enable the receivers to update their one-hop neighbor table for parent select function. The child nodes of a low energy node wait for a period of time to confirm there are no more *ADJUST* messages, and then the child nodes update their neighbor tables. After that, the child nodes can select a new parent via the updated neighbor tables.

Algorithm 6 : Route Maintenance - Forest Adjustment

Input:
G : connected network
N : set of sensor nodes
Output:
Reduce the degrees of the low energy nodes in N

```
for each sensor node n, n ∈ N do
    if the energy level of node X ≤ ENERGY THRESHOLD then
        X broadcasts LEAVE message with ID
    end if
    if LEAVE message L received then
        if hopLevel(n) = hopLevel(X)+1 then
            Remove entry of X in neighbor table
        else if hopLevel(n) = hopLevel(X) then
            Broadcast ADJUST message with ID, energy, and number of child nodes
        end if
    end if
    if ADJUST message A received then
        Update neighbor table
    end if
    Child nodes of X select a new parent
end for
```

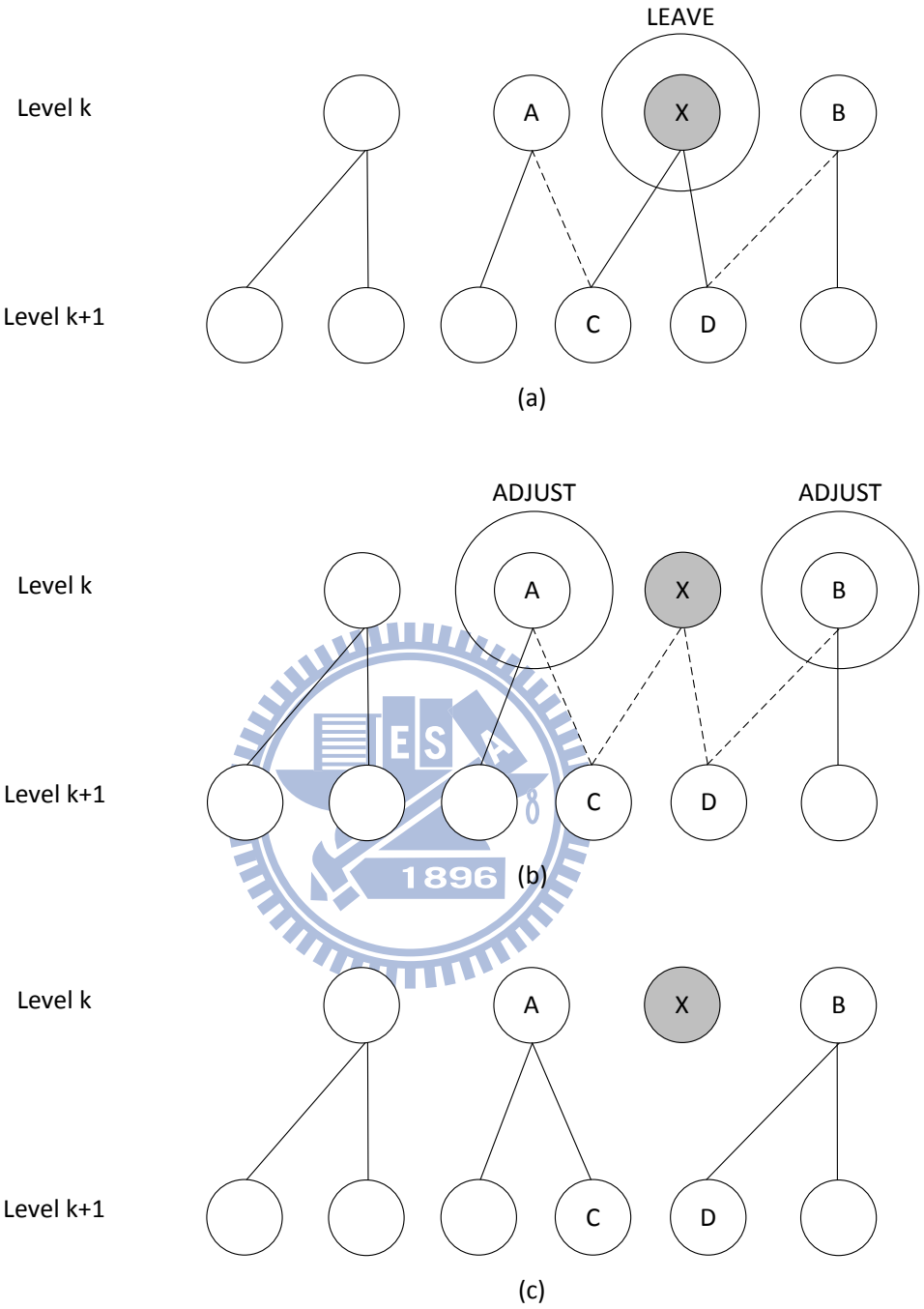

Figure 4.6: Illustration of Algorithm 6 for forest adjustment

Figure 4.6 illustrates the forest adjustment. In Figure 4.6(a), node X broadcasts a *LEAVE* message since its energy is lower than a threshold. After that, there are four neighbor nodes A, B, C, and D receiving the *LEAVE* message in Figure 4.6(b). Node C

and D are located at level k+1 which is lower than the level node X located at by one. Therefore, node C and D remove the entry of node X in their neighbor tables. Node A and B are located at level k which is the same as the level node X located at. Hence, node A and B broadcast *ADJUST* messages with their ID, energy, and number of child nodes. After a period of time, there are no more *ADJUST* messages as shown in Figure 4.6(c). Then, node C and D can select a new parent according to the updated neighbor table. Node C selects node A as the parent, and node D selects node B as the parent. The degree of node X is reduced while the lifetime is also prolonged.

Chapter 5

Simulation Results

In this section, we evaluate the performance of our proposed algorithm via simulations. We assume there are 100 nodes in a wireless sensor network while nodes are spread in a grid topology. The sensor network is located at a square area with side-length 100m. The initial energy of each node is given as 0.1J. Transmission range R is set to 20m. Each node generates $B = 48$ bytes of data in each round. There are four base stations which located at coordinates (25, 25), (25, 75), (75, 25), (75, 75) respectively. Table 5.1 summarizes the simulation parameters. We adopt the simulation parameter above except there is an otherwise specification. In the simulation, we compare the lifetime performance in terms of rounds, control message, and data message of our algorithm between those of Wu's algorithm in [14].

Number of nodes	100
Field	100×100
Initial energy	0.1
B (bytes)	48
R (transmission range)	20

Table 5.1: Simulation parameters

5.1 Impact of Number of Sensor Nodes

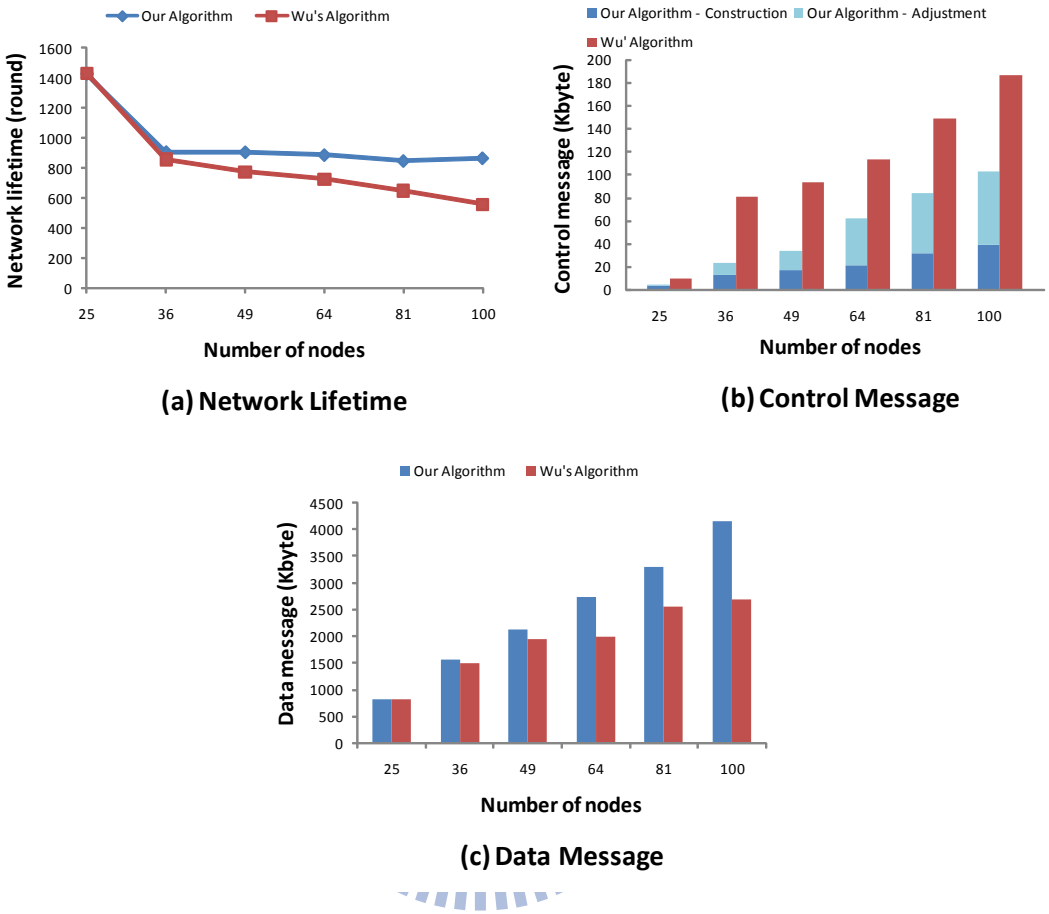

Figure 5.1: Simulation with difference number of nodes

We adjust the number of nodes from 25 to 100 while the topology is the grid. In Figure 5.1(a), the network lifetime decreases while the number of nodes increases. Because when the number of nodes increases, the average degree also increases. In results in average round of each node decreases, so the network lifetime decreases. We can see that the network lifetime of our algorithm is more than that of Wu's algorithm except the node is 25. Since the number of nodes is less, each node has fewer neighbor node. Once the energy of a node is low, there may be no adjustment to enable the child nodes to select a new parent. Hence, the network lifetime is less than that of Wu's algorithm when the number of nodes is less. On the other hand, in Figure 5.1(b), the control messages of Wu's algorithm is more than that of our algorithm since Wu's algorithm is a centralized algorithm. Since the network life time of our algorithm is longer, the data messages are

also more than that of Wu's algorithm as shown in Figure 5.1(c).

5.2 Impact of Number of Base Stations

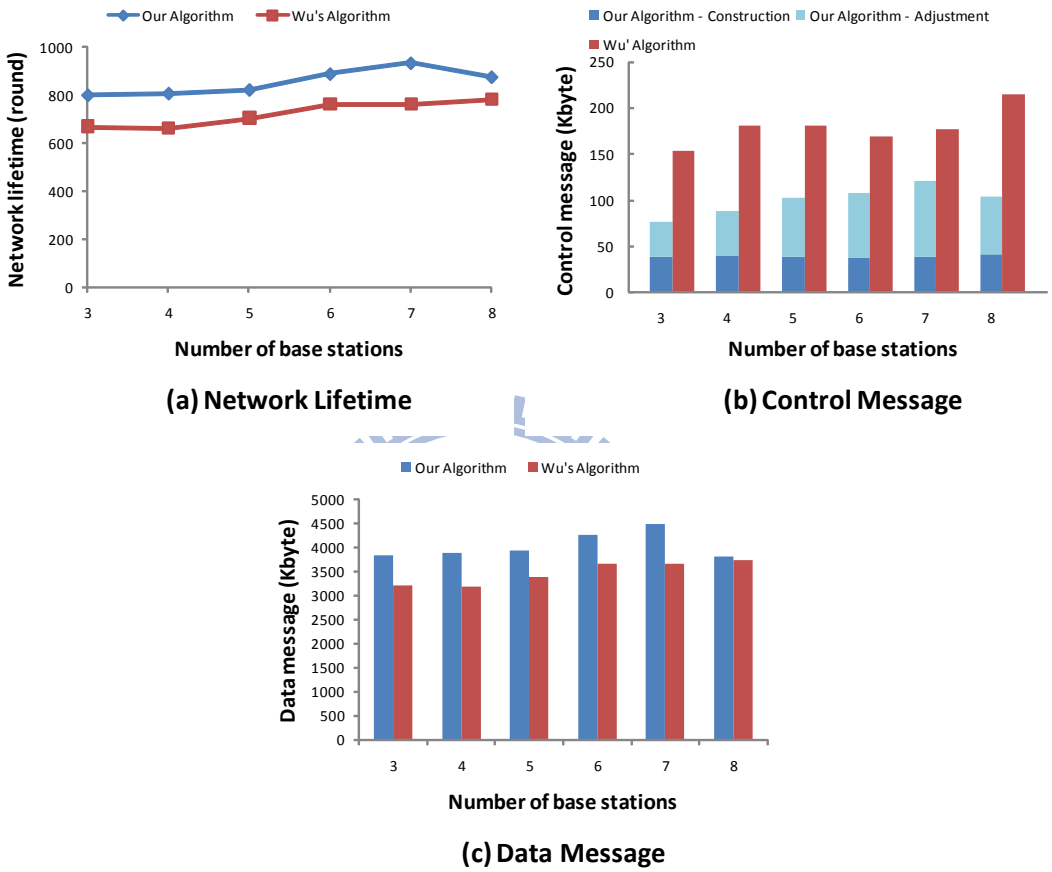

Figure 5.2: Simulation with difference number of base stations

Number of base stations	Location
3	(30, 30), (60, 30), (45, 60)
4	(25, 25), (75, 25), (25, 75), (75, 75)
5	(25, 25), (75, 25), (25, 75), (75, 75), (50, 50)
6	(25, 30), (50, 30), (75, 30), (25, 60), (50, 60), (75, 60)
7	(50, 15), (25, 30), (75, 30), (50, 45), (25, 60), (75, 60), (50, 75)
8	(40, 20), (80, 20), (20, 40), (60, 40), (40, 60), (80, 60), (20, 80), (60, 80)

Table 5.2: Location of base stations

Then, we vary the number of base stations from 3 to 8. Table 5.2 shows the location of each base station while the number of base stations varies. Obviously, regardless of the number of base stations, the network lifetime of our algorithm is longer than that of Wu's algorithm as show in Figure 5.2(a). When the number of base stations increases, the network lifetime also increases. Since the number of data aggregation tree rooted at each base station increases, it reduces the average degree of each node. Therefore, the network lifetime is prolonged while adding the number of base stations. In Figure 5.2(b), the control messages exchanged between each base stations of Wu's algorithm increase since the number of base stations increases. On the other hand, the control messages of our algorithm are about the same.

5.3 Impact of Transmission Range

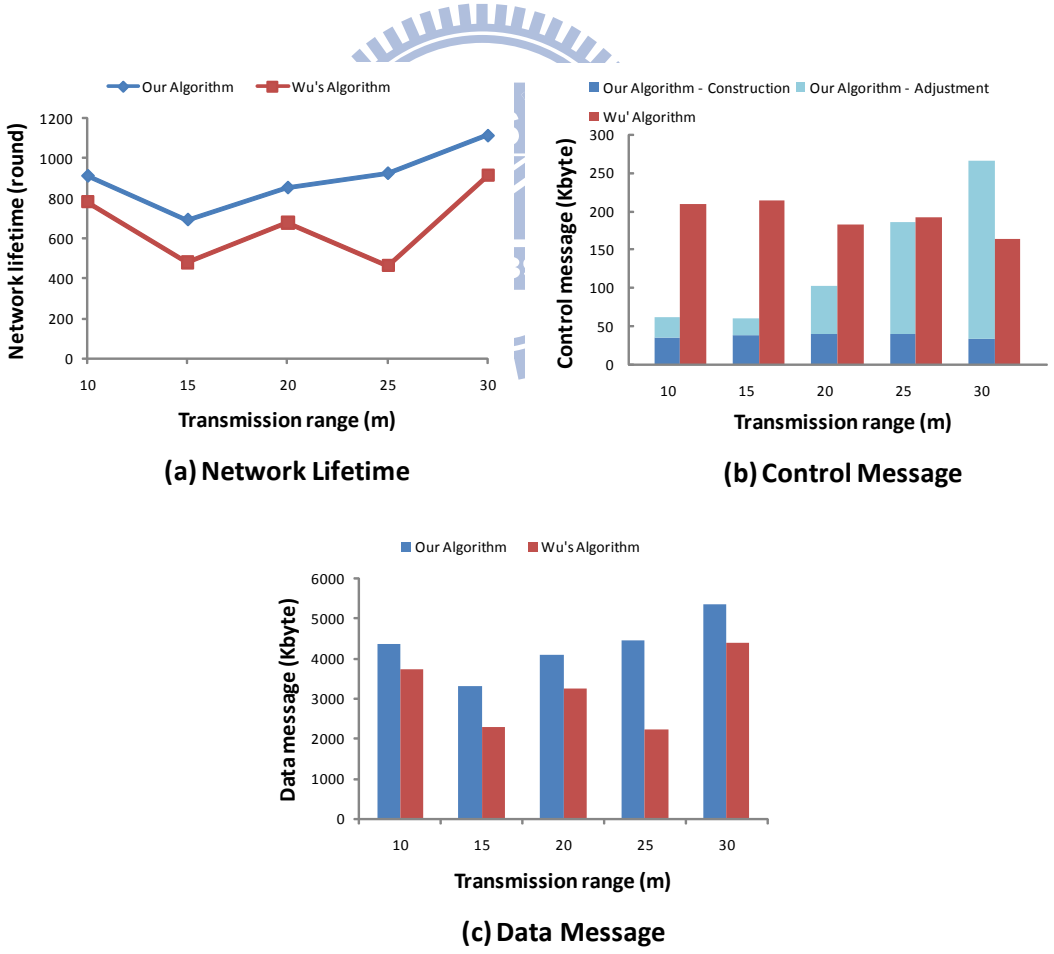

Figure 5.3: Simulation with difference transmission range

We adjust the transmission range from 10m to 50m. In Figure 5.3(a), we can see that our algorithm surpasses Wu's algorithm in the network lifetime. It is owing to the forest adjustment to prolong the network lifetime. As shown in Figure 5.3(b), when the transmission range is less than 20 m, the control messages of our algorithm are less than that of Wu's algorithm. While the transmission range is more than 30 m, the control messages of our algorithm are more than that of Wu's algorithm. Since the transmission range is large, each node can have more neighbor nodes. It means that every node have plenty of potential parent nodes. When the energy level of a node is low, the child nodes of low energy node very likely change its parent to reduce the degree of low energy node. It results in a large amount of message exchange of forest adjustment. However, forest adjustment prolongs the network lifetime effectively.

5.4 Impact of Network Area Size

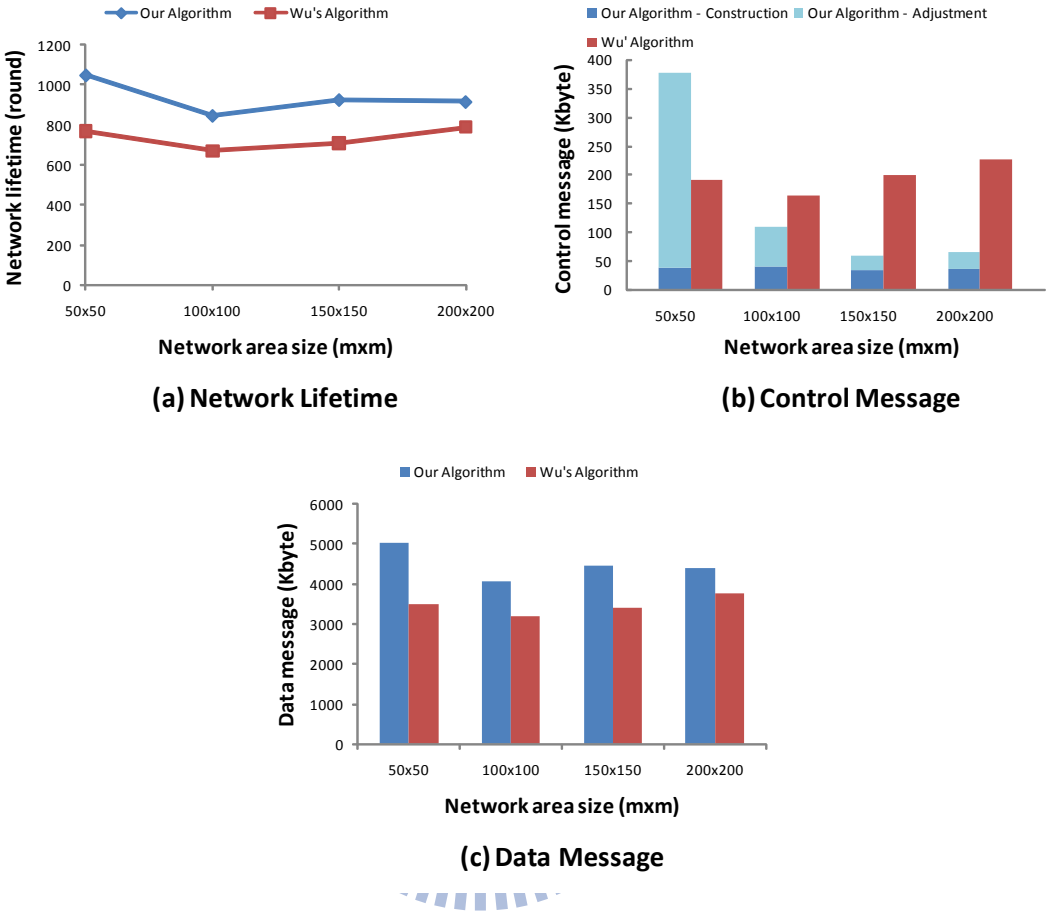

Figure 5.4: Simulation with difference network area size

We increase the network area from 50m × 50m to 200m × 200m. In Figure 5.4(a), no matter how the network area size changes, the network lifetime of our algorithm still exceeded that of Wu's algorithm. When the network area size is small, the density of sensor nodes is high. That is each node has more neighbor nodes in small network area. As shown in Figure 5.4(b), the control messages of our algorithm is more than that of Wu's algorithm while the network area size is 50m × 50m. The more neighbor nodes results in the more message exchange of forest adjustment. Once the network area size is larger than 100m × 100m, our control messages are less than that of Wu's algorithm.

We observe the network lifetime performance for the cases above, the network lifetime achieved by our algorithm outstrips Wu's algorithm. Our distributed algorithm can scale up while the number of nodes increases in a sensor network. Although the message

overhead may increase while the neighbor nodes increase, that is because of message exchange of forest adjustment. The adjustment mechanism effectively prolongs the network lifetime.

Chapter 6

Conclusion

In this paper, we study the problem of constructing a maximum network lifetime data aggregation forest. The problem is proven to be NP-complete. Therefore, we propose a distributed and efficient algorithm to construct the data aggregation forest. The nodes with more energy should also keep more child nodes in order to prolonging the network lifetime. Each node can determine a parent node only requires local knowledge based on the concept. After forest construction, route maintenance is also considered. If there is node failure or addition in a sensor network, the algorithm can reconstruct the topology locally and efficiently. Moreover, for increasing the network lifetime, we present a mechanism to reduce the degree of the node while the energy is insufficient. Finally, our simulation results show that the proposed solution is efficient and prolong the network lifetime as long as possible. While increasing the number of nodes, our algorithm can also scale up to prolong the network lifetime. When the density of sensor node is low, the message exchange of our algorithm also decreases. In conclusion, our algorithm performs well in large scale sensor networks while the density of sensor nodes is not high.

Bibliography

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

[2] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan. Upper bounds on the lifetime of sensor networks. In *Proceedings of the IEEE International Conference on Communications*, pages 789–790, 2001.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the IEEE Hawaii International Conference on System Sciences*, pages 3005–3014, 2000.

[4] M. Khan, G. Pandurangan, and V. S. A. Kumar. Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks. *IEEE Transactions on Parallel Distributed Systems*, 20:124–139, January 2009.

[5] N. Li, J. C. Hou, and L. Sha. Design and analysis of an MST-based topology control algorithm. In *Proceedings of the IEEE INFOCOM*, pages 1702–1712, 2003.

[6] X.-Y. Li, X.-H Xu, S.-G. Wang, S.-J. Tang, G.-J. Dai, J.-Z. Zhao, and Y. Qi. Efficient data aggregation in multi-hop wireless sensor networks under physical interference model. In *Proceedings of the IEEE International Conference on Management and Service Science*, pages 353–362, 2009.

[7] J. Liang, J. Wang, J. Cao, J. Chen, and M. Lu. An efficient algorithm for constructing maximum lifetime tree for data gathering without aggregation in wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, pages 506–510, 2010.

[8] S. Lindsey and C. S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Proceedings of the IEEE Aerospace Conference*, pages 1125–1130, 2002.

[9] R. Rajagopalan and P. Varshney. Data-aggregation techniques in sensor networks: a survey. *IEEE Communications Surveys and Tutorials*, 8:48–63, 2006.

[10] H. O. Tan and I. Korpeoglu. Power efficient data gathering and aggregation in wireless sensor networks. *ACM SIGMOD Record*, 32:66–71, December 2003.

[11] H. O. Tan, I. Korpeoglu, and I. Stojmenovic. Computing localized power-efficient data aggregation trees for sensor networks. *IEEE Transactions on Parallel Distributed Systems*, 22:489–500, March 2011.

[12] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12:231–268, 1980.

[13] Y. Wu, S. Fahmy, and N. B. Shroff. On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm. In *Proceedings of the IEEE INFOCOM*, pages 356–360, 2008.

[14] Y. Wu, Z. Mao, S. Fahmy, and N. B. Shroff. Constructing maximum-lifetime data gathering forests in sensor networks. *IEEE/ACM Transactions on Networking*, 18:1571–1584, October 2010.

[15] L. Zhang, S. Chen, Y. Jian, and Y. Fang. Distributed progressive algorithm for maximizing lifetime vector in wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, pages 2410–2418, 2009.