Sensor Data Aggregation and its Allocation in Cloud Environments

March 2016

Yaling Tao

Sensor Data Aggregation and its Allocation in Cloud Environments

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2016

Yaling Tao

Acknowledgments

First and foremost, I would like to express the deepest respect and most sincere gratitude to my advisor, Prof. Yongbing Zhang, for his guidance, patience, and support. I feel inspired by his boundless enthusiasm, dedication to excellence, careful attention to detail, and infinite patience. I feel privileged to have the opportunity to study under his guidance.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Maiko Shigeno, Prof. Ying Miao, Prof. Akiko Yoshise, and Prof. Yusheng Ji, not only for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I also thank all of the members of Zhang Laboratory for creating an enjoyable environment for research and study. I thank all the people who give me help and assistance in Japan. Finally, I thank my parents and my family for their constant support and encouragement throughout my life.

Abstract

Wireless sensor networks (WSNs) have been gaining increasing attention recently in areas such as environmental monitoring, industrial process monitoring and control, structural health monitoring, etc. Furthermore, with the development of cloud computing, the inherent limitations of WSNs, such as storage capacity and computing capability, can be alleviated by integrating WSNs into the cloud environment. Both a larger market and more opportunities have arisen for WSNs. This thesis is about the study on sensor data aggregation in a wireless sensor network and sensor data allocation in the cloud environment. The former focuses on designing an energy-efficient data aggregation protocol that yields better power efficiency and coverage preservation for a large-scale wireless sensor network, while the latter focuses on implementing an optimal data allocation approach that minimizes the on-demand data access and maintenance costs in a sensor cloud environment.

In this thesis, a *flow-balanced routing* (FBR) protocol is proposed for a large-scale sensor network that attempts to achieve both power efficiency and coverage preservation. In FBR, the sensors randomly deployed in the target field are firstly grouped into clusters and a cluster head is determined in each cluster on the basis of the overlapping degree of each sensor which is defined as the ratio of the overlapping area with other sensors to the whole sensing area of the sensor. Then, a multi-level backbone with the base station at the top level and the cluster heads at lower levels is constructed, where a node can be allowed to have more than one parent in the upper level so that the data can be transferred through multiple paths, and the energy consumption among the sensors can be evenly balanced. The network topology also can be reconfigured with low cost by locally repairing the network only at the locations where some cluster heads run out of their energies and drop out of the backbone. The FBR protocol has been evaluated in comparison with previous ones by simulation. The simulation results show that FBR yields much longer lifetime and better coverage preservation than previous protocols.

Data gathered from various sensor networks are generally stored at different data centers which are distributed across the Internet. A user can access to any data he/she

needs but may take long time if the required data is stored at a faraway data center. In this thesis, a data cache approach is proposed wherein some data cache nodes are placed in the network to store the data needed by their nearby users to reduce the data access cost. Three kinds of costs for data access and maintenance can be considered: 1) the assignment cost for transferring the data items periodically from their data sources to the cache nodes, 2) the placement cost for storing the data items at the cache nodes, 3) the access cost for the users to obtain their required data. In order to minimize these data operation costs, three data allocation problems are formulated: a single-type data allocation problem (SDAP), an uncapacitated multi-type data allocation problem (UMDAP), and a capacitated multi-type data allocation problem (CMDAP). Furthermore, an algorithm extended from the Lagrangian relaxation approach, a greedy caching algorithm, and an efficient heuristic algorithm are proposed to solve these problems. Our proposed algorithms are examined by numerical experiments and yield much better performance than other approaches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A wireless sensor network (WSN) is composed of wireless sensors (simply called sensors later in this thesis) each of which can communicate with one or more than one other sensors over a wireless communication channel. The data sensed by the sensors in the network are commonly collected to a data center called a base station or a sink. The use of WSNs has originally been proposed for military applications such as battlefield surveillance and recently has been expected to play important roles in various kinds of industrial and consumer applications. A WSN can be used in an agricultural application to monitor nutrients in the soil, humidity, temperature and sunlight around the target fields [1], or in a risk monitoring application in hostile environments to track mountain landslide or chemical gas leak [2].

One of the most critical constraints of a WSN is the power supply since it is generally difficult to provide the wired power supply for sensors in harsh environments and instead a battery-driven approach has to be taken [3]. The power consumption of a sensor is mostly due to data sensing, processing, and transmission, and the data transmission occupies the majority. Therefore, it is important to have an efficient power-saving way to transmit data between sensors.

Many protocols have been proposed to improve the energy efficiency for a WSN in the past decade. Almost all of the protocols can be classified as clustering and

multi-hop transmission according to the network structure. Clustering protocols aim at grouping sensors into clusters so that cluster heads can do some aggregation and reduction of data and then transfer data to the sink directly. The cluster heads, especially those located far away from the sink in a large-scale network, may run out of batteries quickly. Usually, the network is reconstructed frequently so that different sensors can be selected to play the cluster head roles. Reconstructing a network frequently will lead to a large amount of overhead. Multi-hop transmission protocols aim at transferring data via multiple hops to reduce the energy consumption caused by long-distance transmission. Most of existing multi-hop transmission protocols are committed to constructing a tree. However, the workload imbalance between sensors is an inherent drawback of a tree topology network. How to efficiently collect the data from the sensors to the sink so as to prolong the battery lifetimes of sensors is still one of the most important issues for WSNs, and we first focus on this issue in this thesis.

In addition, the sensor data collected from a WSN is traditionally provided to the users via the sink of the network and the data access is reluctantly limited to the users who have direct connections to the sink. Recently, the concept of *sensor-cloud* is proposed to leverage powerful cloud computing technologies to provide excellent data scalability, rapid visualization, and user programmable analysis. The massive computing and storage capabilities of cloud computing can compensate for the limitations of WSNs and enable much more efficient data services [4, 6]. With the sensor-cloud platform, data collected from WSNs can be efficiently maintained in a cloud and easily provided to various users.

The data center of a sensor-cloud can act as a data source providing various data to the users in a continuous, pervasive, and real-time manner. However, the data centers are usually distributed sparsely over the Internet and users far away from a data center may experience long access delay. Therefore, it is important to have an efficient way to provide data services to the users, especially when the continuous or real-time data are needed in a data-oriented application for the environment monitoring, health-care or position-tracking.

A few studies on the sensor-cloud have been done recently. However, almost all of those works focus on the concept elaboration about the overall system construction rather than the specific sensor data management. Even though data caching is an efficient way to speed up the data fetch, existing data caching strategies can not be used directly in the sensor-cloud environment. The common strategies where data is only considered to be cached at the local server have limitations, such as copies of the same data may occupy some neighboring servers while other equally important data can not be cached due to the limited storage space. The resource sharing and cooperation between servers need to be considered to improve the performance of data services. A number of cache nodes near to the users, e.g., to equip the network edge routers with data caches, can be allocated to form a cache network. How to allocate data copies at these cache nodes so as to minimize the data operation costs is one of the important issues for providing efficient data services, and that is another main topic we focus on in this thesis.

1.2 Objectives of This Thesis

In this thesis, we focus firstly on the data aggregation in a WSN and then on the data allocation in a sensor-cloud with the following objectives.

1. We propose an energy efficient data aggregation protocol to alleviate the power limitation for wireless sensor networks. Following this protocol, the sensors deployed over the given target field can spontaneously construct a network, and then the sensed data can be conveyed to the sink efficiently. The main contents of this protocol need to include: the network construction approach, the selection criteria of transmission paths, the cooperation ways between sensors, the power management of sensors, ect.

2. We design an efficient data allocation approach to improve the performance of data services in the sensor-cloud environment. Following this approach, sensor data can be accessed at any time and from anywhere. The main consents of this approach need to include: the construction of data service model, the allocation

strategy of cache severs, the analysis of factors involved in system costs such like system characteristics, data characteristics, and users characteristics. And most of all the data access and operation costs should be minimized.

1.3 Contributions

Firstly, a flow-balanced routing (FBR) protocol for data aggregation in WSNs is proposed in this thesis. Compared with previous protocols, the FBR yields more than twice the lifetime of network. The advantages of FBR can be summarized as follows.

- FBR tackles the load imbalance problem inherent in the tree topology by conveying data over a novel multi-level backbone. Sensors are first grouped into clusters then the sink and the cluster heads build a backbone with the sink at the top level and the cluster heads at lower levels. The network topology may not be a tree structure, where each cluster head may have multiple parents and all the parents of a head should be at the same level which is one higher than the head. In this way, each cluster head can transfer its data to the sink through multiple paths to the sink, and the flow from each head to the sink can be distributed to its parents, resulting in energy balance between sensors. As multiple paths are constructed, the network robustness also can be ensured without rebuilding the network frequently.

- FBR alleviates the overhead problem in previous works by repairing network locally instead of rebuilding the network thoroughly. Compared with previous work in which the network is reconstructed in every data aggregation round, the FBR performs the whole network clustering and construction only once at the beginning of the network operation. The network repair is started only when a cluster head runs out of its energy and only in a place where the exhausted head drops out of the backbone.

- FBR ensures energy efficiency as well as good coverage preservation for network

by taking into account the overlapping degrees of sensors in the clustering decision. The overlapping degree of a sensor is defined as the ratio of the overlapping area with other sensors to the whole sensing area of the sensor. A sensor with the largest overlapping degree within a predefined cluster range priors to be the cluster head. Those sensors whose sensing areas wholly overlap with others are put into sleep mode to save energy.

Secondly, an efficient data allocation strategy is proposed to improve the sensor data services experience for users in sensor clouds. We construct a network using cache nodes at access networks to cache and process the data generated by sensor networks. Users can obtain the data they required directly from the base station that stores this data or from any cache node that contains a copy of the data. We construct a sensor data service model for sensor clouds. System characteristics such as the network topology and storage capacity of each server node, data characteristics such as the data types, data sizes, data update interval, the demands of users for data, and costs involved in assigning data periodically from data sources to data server nodes, storing data at the server nodes, and transferring data from server nodes to users are considered. The main contributions we have in this thesis to the data allocation in a sensor cloud can be summarized as follows:

- We solve the single-type data allocation problem (SDAP), in which only one sensor network is integrated into the cloud, by designing an algorithm extended from the Lagrangian relaxation approach and a greedy caching algorithm. Compared with the Lagrangian relaxation algorithm which is committed to find a feasible solution between the decidable upper and lower bounds, the greedy algorithm decides whether or not locate a data copy at the server node based on the current situation in a simpler and easier way. From the results of numerical experiments, the greedy caching approach yields a similar solution with the Lagrangian relaxation one.

- We solve the uncapacitated multi-type data allocation problem (UMDAP), in which multiple sensor networks are considered and each of cache node in the

cloud is no capacity limitation. The Lagrangian relaxation algorithm and greedy caching algorithm are adopted to get a solution and maintain the same performance as the SDAP.

- We solve the capacitated multi-type data allocation problem (CMDAP), where multiple sensor networks and capacity limitations of cache nodes are considered, by combining algorithms for SDAP with another heuristic algorithm. The performance is examined by numerical experiments and the results show that the performance difference between our proposed algorithms and an ideal case where each cache node has no capacity limitation is negligibly small.

1.4 Organization of This Thesis

The rest of this thesis is organized as follows. Chapter 2 introduces the basic concept of sensors and wireless sensor networks, then summarize some important related works in aspects of data aggregation technologies, sensor clouds, and data services technologies etc. Chapter 3 describes the proposed FBR protocol in detail, including the network model, proposed algorithms, performance evaluation and discussion, and the brief conclusions of data aggregation for WSNs. Then the next chapter presents the data allocation strategies for sensor clouds. The corresponding data service model, optimization formulation and algorithms for data allocation problems, performance evaluation and discussion are also given in this chapter. Finally, chapter 5 concludes this thesis with a summary and gives future research directions.

Chapter 2

Related Work

2.1 Sensors

A sensor is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena. The output is generally a signal that is converted to human-readable display at the sensor location or transmitted electronically over a network for reading or further processing[7]. Advances in miniaturization and low-power design have enabled the development of extremely small and low-cost sensors that possess sensing, data processing and transmission capabilities. Sensors are used to monitor a wide variety of ambient environments such as temperature, pressure, humidity, soil makeup, vehicular movement, noise levels, lighting conditions, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects [8]. A sensor is generally made up of four basic components as shown in Figure 2.1: a sensing unit, a processing unit, a transceiver unit, and a power unit [3]. The sensing unit is responsible for observing a surrounding physical phenomenon and converting the phenomenon into a digital signal, while the processing unit is undertaking the data processing, such as data compression. The transceiver unit consists of a pair of a transmitter and a receiver and is responsible to communication with other devices, and the power unit supports energy for the whole node. All of these units are fit into a matchbox-sized module [9]. Wireless sensors have

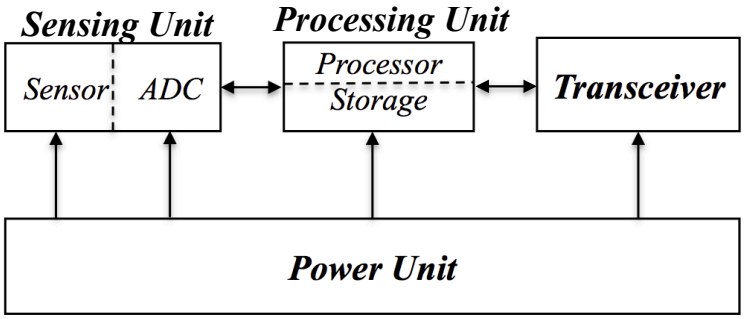


Figure 2.1: The components of a sensor node [3].

many advantages over the wired sensors due to their easy deployment and network scalability. In this thesis, we only consider sensor networks constructed by wireless sensors. The power unit of a wireless sensor is usually supported by a battery with limited energy capacity and therefore the power becomes a scarce resource accordingly. Sensors can be used for continuous sensing, event detection, event ID, location sensing and local control of actuators. This thesis only considers to use sensors for continuous sensing, like humidity measure in forest, landslide detection, and so on. Such a kind of applications usually needs a large number of sensors to cover the given service areas.

2.2 Wireless Sensor Networks

A wireless sensor network (WSN) is a network formed by a large number of sensors, which are deployed throughout a given target field. The sensors are usually deployed densely, within tens of feet of each other, to ensure a good coverage. Sensors can be either thrown in mass or placed one by one. After deployment, additional sensors also can be re-deployed to replace the disabled sensors, which happens often in an inaccessible and large-scale field. Due to the advancements of technology and sensors getting smarter, smaller, and cheaper, WSNs have been adopted in numerous applications which can be categorized into several domains as follows [3]:

- Military applications, such as battlefield surveillance, reconnaissance of opposing forces and terrain, targeting, battle damage assessment, nuclear, biological and

chemical attack detection and reconnaissance, and so on.

- Environmental applications, such as forest fire detection, biocomplexity mapping of the environment, flood detection, and precision agriculture.
- Health applications, such as tele monitoring of human physiological data, tracing doctors and patients inside a hospital, and drug administration in hospitals.
- Home applications, such as home automation, smart environment.
- Other commercial applications, such as environmental control in office buildings, interactive museums, detecting and monitoring car thefts, managing inventory control, vehicle tracking and detection, and so on.

In most applications, data sensed by the sensors are sent to a base station, usually called the *sink*, directly (in single hop) or via multiple intermediate sensors (in multi-hop). Note that there may also be multiple sinks in a WSN, since the main concern in this thesis is how to efficiently collect data from sensors to the sink, it is sufficient to consider only one sink in our network model. The sink may be located either inside the sensor field to shorten the transmission distances or beside the sensor field for easier maintenance depending on the applications. An example of WSNs is shown in Figure 2.2 where sensors are randomly located at the target field, which may be a desert, volcano, etc., to monitor ambient conditions, such as temperature. These sensors coordinate among themselves to construct a communication network according to a predetermined protocol. Sensors sense the data periodically and then pass them in cooperation with other sensors to the sink.

Since sensor nodes are power constrained, the power limitation also becomes a critical constraint for WSNs accordingly. Generally, power is mainly consumed by data processing and data transmission, both are depend on the amount of data and the latter is also determined by the transmission distance. It is inefficient for all the sensors to transmit the data directly to the sink. Therefore, it is important to design an energy-aware protocol, which may transmit the sensed data from the sensors in an efficient way to the sink, to control the power consumption.

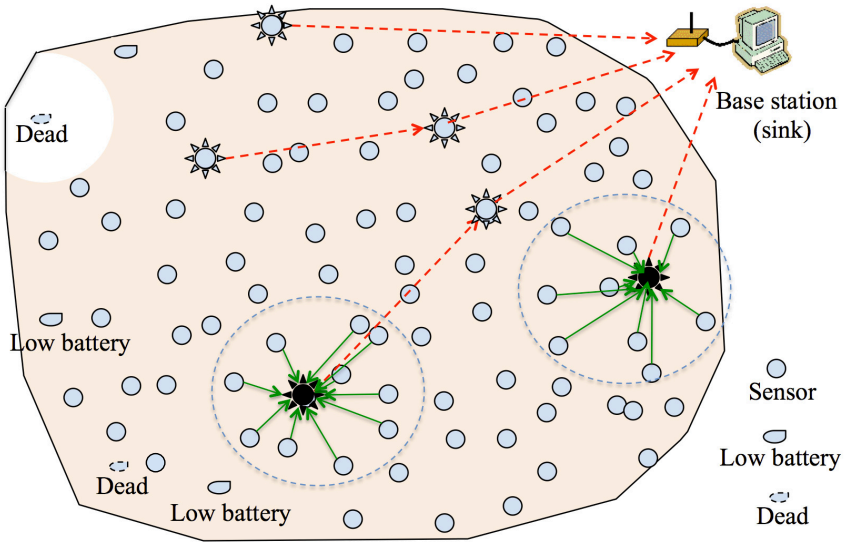


Figure 2.2: Example of sensor network.

Two metrics, *network lifetime* and *coverage lifetime*, are usually used to evaluate the energy-efficiency of a sensor network [10, 11]. Those metrics indicate the time durations from the beginning instant of the network operation to the instant when a given percentage of sensors *die* and when the ratio of the current *coverage*, generated by the active sensors, to the initial coverage, generated by all the sensors, drops below a predefined threshold, respectively. For instance, for a network where all nodes work together is vital, we specify the threshold as 100%, the network lifetime is defined as the number of data aggregation rounds from sensors to the sink until the first sensor dies. Similarly, when the threshold of coverage is set to 100%, the coverage lifetime is the number of data aggregation rounds until the coverage of network begins to decrease. Note that we say a sensor is dead when it does not normally work due to the power depletion, in contrast to that a sensor is alive if it has enough energy to complete normal work, an alive sensor can sense a given range of area called coverage. As shown in Figure 2.2, some sensors suffered larger data transfer volume or longer transmission distance may die first. A uniform energy drainage across the entire network is needed to ensure a longer lifetime. In this thesis, we aim to design a new energy-aware data aggregation protocol that yields longer network lifetime and better coverage preservation for the WSNs with large number of sensors.

2.3 Sensor Data Aggregation

Data aggregation is defined as the process of aggregating the data from multiple sensors to eliminate redundant transmission and provide fused information to the sink. It usually involves the fusion of data from multiple sensors at intermediate nodes and the transmission of the aggregated data to the sink [12]. A number of energy-based data aggregation protocols have been proposed. Techniques such as network clustering, sensor scheduling, and multi-hop transmission are widely used to improve energy efficiency for WSNs. In this subsection, we summarize the related works from these three aspects respectively, and say how our work differs from these works.

2.3.1 Network Clustering

In a large scale wireless sensor network, it is inefficient for sensors to transmit the data directly to the sink. Network clustering can result in significant energy savings for the energy constrained sensors. Clustering is a technique to group several sensors into a cluster with one as the head and the others as the members, shown in Figure 2.3. Each member sends data to the cluster head, and then the cluster head does data pre-processing, such as data compression, for collected data and conveys the aggregated data to the sink over the network.

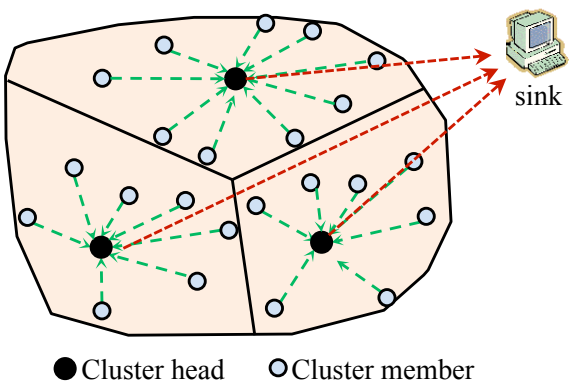


Figure 2.3: Network clustering.

Most previous clustering approaches [13, 14, 15, 16] mainly focus on the head selection and the cluster construction, rather than the coverage preservation and the

data routing after the cluster formation. Cluster heads, especially those heads far away from the base station may suffer a remarkable energy consumption. To balance the traffic flows from sensors to the sink and equalize the residual energy among the sensors, most previous approaches perform the cluster formation and the network construction periodically to select different sensors to take the roles of cluster heads. This way would shorten the lifetime since the overhead cost for transferring the control messages between the sensors cannot be ignored.

Low-energy adaptive clustering hierarchy (LEACH) [13] is a well-known and simple distributed clustering approach wherein each sensor elects itself as a cluster head with a certain probability (for example, 5%), and then broadcasts an advertisement message to the rest of the nodes. Each non-cluster-head node decides the cluster to which it will belong according to the received signal strength of the advertisement. The cluster heads act as routers aggregating and transferring sensing data to the sink directly. In order to balance energy of the cluster-head nodes and the non-cluster-head nodes, the clusters are reorganized in each data aggregation round.

A centralized version of LEACH, called LEACH-centralized (LEACH-C), was proposed in [14]. In contrast to LEACH, each sensor sends its location information along its residual energy to the sink, and then the sink computes and determines the clusters. Furthermore, a sensor with residual energy lower than the average energy of all the sensors can not become a cluster head. LEACH-C can reduce overhead for each sensor to construct clusters in the centralized way, and avoid the sensor with lower energy becoming a head. However, the inherent disadvantages in terms of flexibility and fault tolerance made the centralized computing not be used widely in WSNs.

Another extension, called the hybrid energy-efficient distributed clustering (HEED) approach [15], considers the residual energy of each sensor and attempts to obtain a well distribution of the cluster heads in the service area. The computation time of HEED is extremely long since the probability of becoming a head is computed iteratively depending on the residual energy of each sensor.

Energy efficient clustering and data aggregation (EECDA) protocol [16] applies clustering technique into heterogeneous sensor networks. Based on the idea of LEACH,

EECDA derives three new thresholds of cluster heads election probability for three types of sensors, named normal, advanced, and supper nodes, respectively. After the cluster head election, each sensor selects a path via the optimal head to the sink. The optimal head for a sensor means that the sum residual energy of the sensor and this selected head after this transmission via this path should be higher than any other paths via other heads. This needs to know the residual energy of all heads, and the distance from the current sensor to all heads, and also all heads to the sink.

All the above schemes are single-hop clustering schemes, where each head collects data from its members and then conveys data to the sink directly. This way is not applicable to networks deployed in large regions. Furthermore, the cluster head determination is mainly based on the residual energy of sensors without considering the coverage. Recently, a coverage-based clustering approach, CPCP, was proposed [17]. Two kinds of coverage preservation approaches are proposed. One is based on the *coverage redundancy* defined by the number of sensors at each point, and therefore if the sensing area of a sensor is covered by more sensors, the sensor will have a higher priority to be a cluster head. The other is based on the *coverage energy* defined by the total residual energy that can be used to monitor a location, and therefore if there are more total residual energy that can be used to monitor sensing area of a sensor, the sensor will have a higher priority to be a cluster head. The drawback of this protocol is that it may take much time to calculate the values of the coverage redundancy and the coverage energy. The basic idea of our proposed clustering approach is similar to that of CPCP, but our proposed approach only calculate the ratio of the area of each sensor that overlaps with other sensors, resulting in a simpler mechanism.

2.3.2 Sensor Scheduling

Sensor scheduling is a technique to switch some sensors off to save power while keeping the network connectivity to satisfy a given coverage preservation requirement [17, 18, 19, 20, 21]. An example network using this technique is shown in Figure 2.4. Some sensors are set into sleep mode to save power without lowering the performance of network. When the network topology is broken due to some active sensors run out of

their power, these sleep sensors will turn into active mode and participate the network reconstruction. This technique can be combined into the clustering process, and it usually faces the challenges of network connectivity and coverage preservation.

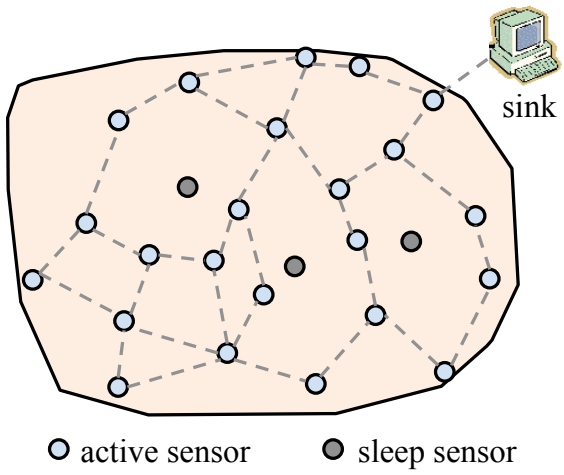


Figure 2.4: Sensor scheduling.

An approach called SPAN [19] was proposed to save power consumption by putting some nodes into sleep mode for ad hoc wireless networks. SPAN attempts to switch off such nodes that do not affect the network connectivity by maintaining the information of the two hop neighbors in real time. However, it is not given how to turn a sleeping node on again if the network connectivity is damaged since some active nodes run out of their energy. Furthermore, since the source and the destination in an ad hoc network are not generally fixed, it is unclear how SPAN can be applied in WSNs. As pointed out by Zhang and Hou in [40], the power saving for a wireless ad hoc network and a wireless sensor network is generally different from each other in terms of the objectives for power saving. That is, the algorithms used for wireless ad hoc networks do not address the issue of sensing coverage. It is therefore difficult to apply SPAN directly for a sensor network.

A coverage configuration protocol (CCP) is proposed and compared with SPAN in [20], CCP tries to achieve a given coverage goal by turning off as many sensors as possible while keeping the network connectivity. Initially, all sensors are in the *active* mode, some of them can switch to the *sleep* mode if the coverage requirements are

met. Over time the degree of coverage decreases below the desired level, some sleep nodes will turn into *listen* mode, and then turn into active mode over a listen timer.

In coverage-based clustering approach (CPCP) [17], power scheduling technique is used after cluster formation. In sensor scheduling phase, sensors with higher coverage cost, which is defined to be inversely proportion to either the total energy of neighboring sensors or the overlapping redundant degree with neighboring sensors, have a better chance of becoming active sensors in the upcoming round. In our proposed approach, only the sensing area of a sensor and the area where the sensor and its neighboring sensors overlap are needed in computation. Furthermore, sensor scheduling is implemented after the cluster formation phase, sensors determined to go into sleep mode will not involve the rest work of cluster formation, such as optimal head selection when receiving multiple head messages.

2.3.3 Multi-hop Transmission

Multi-hop transmission has generally been considered an efficient energy-saving approach for large-scale sensor networks [17, 18, 22, 23, 24, 25, 26, 27, 28, 29, 30], and the tree rooted at the sink is the most commonly used multi-hop topology. Figure 2.5 shows an example of multi-hop network which is constructed into a tree. However, the tree topology has an inherent drawback in that each sensor has only one path to the sink, and therefore the data flow passing through each sensor may be imbalanced, resulting in some sensors run out of their energy quickly.

Multi-hop transmission can be achieved for intra-cluster or inter-cluster data transmission. In the former, members of a cluster can transfer the sensed data to the cluster head through multiple intermediate members [18, 23, 24], while in the latter [22, 25], a backbone network is constructed with the cluster heads. Inter-cluster transmission has been widely used in previous researches. In an inter-cluster transmission approach, a cluster head sends the aggregated data from its members to the sink via multiple intermediate cluster heads. An example of a multi-hop transmission mechanism can be found in the IEEE 802.15.4 standard [26], wherein a personal area network (PAN) coordinator triggers the formation of a cluster-tree and works as the root of

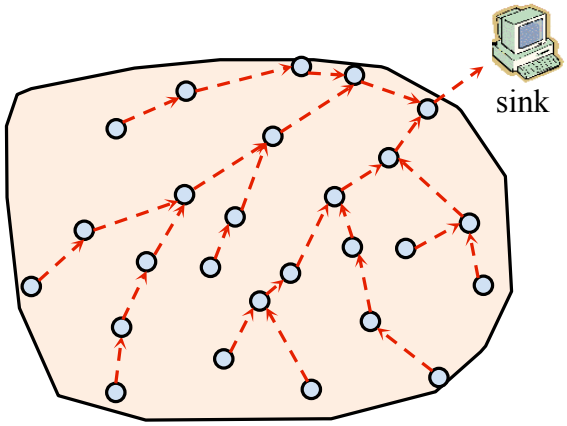


Figure 2.5: Multi-hop transmission to the sink.

the cluster-tree. It broadcasts a beacon message to its neighboring coordinators. A coordinator receiving the beacon decides whether to join the tree and, if it does, it also broadcasts the beacon to its neighbors. However, the standard does not give the details of how to determine the route from each coordinator to the root.

Some recursive approaches are used to construct hierarchical clustering networks [28, 29, 30]. The distributed hierarchical agglomerative clustering (DHAC) approach [30] is a bottom-up network construction scheme wherein some nearby sensors are first grouped into a cluster and a sensor with the smallest identification number is elected as the head. Then, the neighboring clusters are grouped into a larger cluster also with the smallest identification number as the head. This process is repeated until the cluster size reaches a given threshold. The energy-efficient multi-level clustering (EEMC) approach [29] is a centralized and top-down clustering scheme wherein the network topology is constructed from the sink. The sink first collects the location and energy information of all the sensors and then determines the heads on the level next to it and the members of each head. Each head then collects the information of its members and determines the heads on the level next to it again. This process is repeated until the number of levels equals the optimal expected value. Some approaches proposed for single hop transmission such as HEED [15] can be extended to construct a multi-level network by using a recursive approach similar to [28]. We implemented the hierarchical version of HEED, named M-HEED, for comparison. The main problems

in those recursive approaches are that they converge very slowly as the head selection and cluster formation have to be done recursively in each level.

Since most multi-hop transmission approaches are based on a tree topology, the traffic flow passing through the sensors may be unbalanced [31]. To alleviate the flow imbalance problem, some approaches [32, 33, 34, 35] try to find and use alternative tree structures for data transmission. However, they face the problem of how to find and when to use the alternative trees and most importantly they cannot resolve the problem of the flow imbalance. In our initial work [36], we proposed a flow-balanced protocol that constructs the network in multiple levels and in which the network topology may not be a tree structure. Therefore, each head may have multiple paths to the sink and, by balancing the traffic flow on each path, can equalize the energy consumption of each head. Furthermore, we propose a new cluster formation approach that preserves the network coverage in a simple but efficient way. In cluster formation, a sensor with a larger overlapping degree is selected as a cluster head with a higher priority. An efficient scheme is also proposed to reduce the power consumption of a sensor in sleep mode.

Data aggregation for WSNs mainly focuses on the data collection from sensors to the base station, the subsequent work, such as data storage, maintenance, utilization etc., needs to be considered in other ways.

2.4 Sensor Data Services

Though the research in the field of WSNs has been actively carried out in the past decades. Sensor data services are usually limited to a small group of users due to the lack of share-ability, efficient maintenance, and elasticity. New data management technologies need to be considered to increase the value of sensor data. In this subsection, we list related works about the sensor-clouds, data caching technologies, and the traditional facility location problems which share some similarities with our data allocation problems, respectively, and give the differences between our work and those related works.

2.4.1 Sensor Clouds

A sensor cloud is a unique sensor data storage, visualization and remote management platform that leverages powerful cloud computing technologies to provide excellent data scalability, rapid visualization, and user programmable analysis [4]. The image of a sensor cloud is shown in Figure 2.6. Several sensor networks are combined into a cloud platform, data collected from each sensor network can be efficiently maintained on the cloud and easily shared among various users, and the massive computing and storage capabilities of cloud computing can compensate for the limitations of WSNs. With the cloud technologies, data management costs can be reduced significantly [5]. Sensor clouds that integrate various sensor networks into the cloud enable the sensor data to be utilized on a specific infrastructure by visualizing the physical sensors on a cloud computing platform. Data can be offered as services when required by end users via proxy servers. Combining the concept of wireless sensing with cloud computing makes WSNs more attractive. Meanwhile, WSNs act as data sources providing various data to the cloud in a continuous, pervasive, and real-time manner.

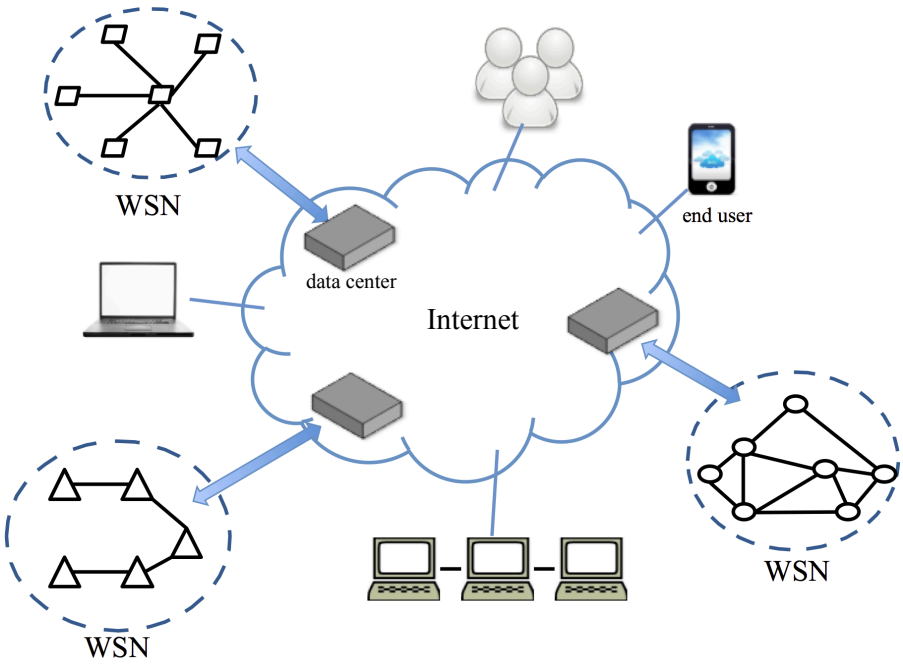


Figure 2.6: An example of sensor-cloud.

Most recent studies [4, 6, 37, 38, 39, 46] related to the sensor-cloud have primarily

focused on the system architecture. Hassan et al. [37] proposed a pub-sub-based model that simplifies the integration of sensor networks with cloud-based community-centric applications, while Mitton et al. [38] focused on designing a pervasive infrastructure where next-generation services interact with the surrounding environment, thus creating new opportunities for contextualization and geo-awareness. Kurz et al. [46] discussed the benefits of sensor-clouds compared to conventional sensor data management and handling in activity recognition systems in an opportunistic manner. Alamri et al. [6] surveyed the sensor-cloud in terms of architecture, applications, and approaches.

Compared with those works focusing on the conceptual explanation of the sensor-cloud system construction, our work focuses on the specific problem of how to allocate the copies of sensor data in the edge networks in order to improve sensor data services by minimizing the total costs for data access and maintenance.

2.4.2 Data Caching

As shown in Figure 2.6, it is not an efficient way that a remote user directly accesses sensor data from data center or the sink of each WSN every time. The data access latency is usually an critical metric for data-intensive applications [47]. It is also unlikely that the proxy with limited capacity resources can download all of the latest data for the end users. Caching data at the edge network near to the end users can alleviate the problems, and therefore it is important to have an efficient way to place the data copies of the data centers at the locations near to the users.

Data Caching is a technique of storing frequently used data at the local server temporarily, so that, when the same data is needed next time, it could be directly obtained from the local server instead of being requested from the far away data source. Data caching can be utilized to improve the data access performance. However, besides of the data access cost for the users, the cached data should be transferred from their original locations and for the real-time data the update costs at the cached servers may not be neglected. Data cache strategies have been studied in [48, 49, 50, 51, 52, 53, 54]. Findings have shown that the number and location of replicas of distinct data items

in a cloud have a strong impact on system performance.

In the collective caching approach proposed by Liao *et al.* [49], the cached data at a cache server can only be accessed by the users connected to the server directly. In the previous approaches proposed for web applications [50, 51], even though a user can obtain the required data from a cache server that is not the nearest one but the data allocation decision is not based on a system-wide view such that the overall data access and maintenance costs may not be minimized. Furthermore, the data items are assumed to be static in nature, that is, they do not need to be updated.

Tan et al. [52] addressed the problem of content placement in peer-to-peer systems with the objective of maximizing the utilization of peers uplink bandwidth resources. In this work, the finite storage capacity per node is not considered as a bottleneck. Data items are classified into three different classes, *Hot*, *Warm*, and *Cold*, according to their popularity rankings. Hot items are cached at all nodes, Warm items are cached at a fraction of nodes, and Cold items are not cached at all. Bjorkqvist et al. [53] also partitioned items into three classes: *Gold*, *Silver*, and *Bronze*. They considered a hybrid content distribution system consisting of a central server storing all of the items and a network of edge nodes caching some of the items. With the objective of minimizing retrieval latency, Gold items are always stored at the edge node, while Bronze items are never stored and Silver items are managed locally either by a collaborative LRU scheme or by a random discarding scheme. Gao et al. [54] also proposed a cooperative caching scheme in which data is intentionally cached at a set of network central locations, each of which correspond to a group of mobile nodes that can be easily accessed by other nodes in the network.

These approaches considered only the data popularity in cache allocation decision but ignored the network topology even though the topology has significant impact on the performance. In our proposed strategy, all the costs for data access, data transmission, and data update are taken into account and our objective is to minimize the total cost for data operation and maintenance.

2.4.3 Facility Location Problem

Our data allocation problem considers to allocate a number of cache nodes near to the users to form a cache network to improve data services. It shares some similarities with the traditional facility location problem (FLP) [55, 56, 57] in operations research and computational geometry concerned with the optimal placement of facilities to minimize costs while considering factors like avoiding placing hazardous materials near housing, and competitors' facilities. In a FLP, a set of facilities with facility-opening costs and a set of clients with demands are given, and the objective is to open facilities and assign clients to the open facilities in order to minimize the facility-opening and client-assignment costs. A facility in FLP can be considered as a data item in our data allocation problem. Furthermore, the costs for facility-opening and for client-assignment can be represented by the costs for data placement and for data access, respectively.

However, there are some fundamental differences between DAP and FLP due to the characteristics of digital data. The data collected from a sensor network is generally updated much often than a facility in FLP and the data should be transferred from the data center across the network to the cached node(s). Therefore, that data transmission cost should be taken into account in the data allocation decision. Furthermore, the computation time for data allocation is more critical than facility location decision in FLP since the user demand may fluctuate more quickly and data lifespan may be much shorter than a facility. Additionally, the capacity of a cache node plays a key role in DAP.

Chapter 3

Data Aggregation in Wireless Sensor Networks

3.1 Network Model

We consider only one sink and a set of homogeneous sensors, denoted by S, that are deployed randomly over the target field. The target field is indicated as $M \times N$ square units. It is assumed that the sink can reach all the sensors in the target field and has no energy limitation. Each sensor has a given unique identification number and a limited sensing range, denoted by r, which covers a disk area centered at this sensor with radius r as shown in Figure 3.1. The data sensed by a sensor can be transferred to the sink directly in single hop or via multiple intermediate sensors. The transmission range of a sensor, denoted by d, can only be tuned to one of the discrete distances $kR\,(k = 1, 2, \ldots)$, i.e., $d = kR$ where R denotes a given fixed distance called the *cluster range* in this thesis. Generally, R is larger than or equal to r and the transmission range d is shorter than the distance from a sensor to the sink.

The sensors within the cluster range of sensor i, R, are called the *neighbors* of sensor i, denoted by N_i. On the other hand, the sensors located in the area with the distance less than $2r$ from sensor i are called the *friends* of sensor i, denoted by F_i, and the sensing areas of sensor i's friends may overlap with that of sensor i. Since sensors are densely deployed in the target field, the sensing area of a sensor may commonly

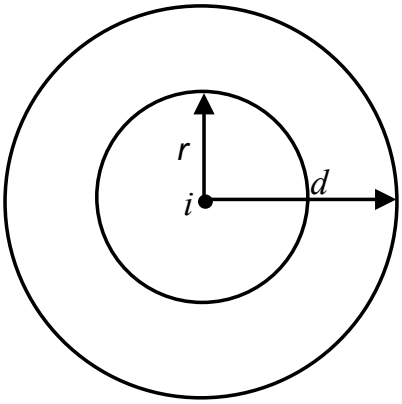


Figure 3.1: The sensing and the transmission areas of sensor i ($R = 2r, k = 1$).

overlap with those of other sensors. The *overlapping degree* of sensor i, denoted by ρ_i, is defined as the ratio of the overlapping area of sensor i with its friends to its whole sensing area as follows.

$$\rho_i = \frac{1}{A_i} \bigcup_{j \in F_i} A_i \cap A_j, \tag{3.1}$$

where A_i denotes the sensing area of sensor i and $A_i \cap A_j$ denotes the area sensor i overlaps with its friend j. Obviously, we have $0 \leq \rho_i \leq 1$, and when $\rho_i = 1$ it means that the sensing area of sensor i is totally covered by its friends. Figure 3.2 illustrates an example in which sensor i has two friends, j and k, and the area sensor i overlaps with sensors j and k is colored gray. Some methods to calculate the overlapping area of multiple sensors are detailed elsewhere [18, 20, 21, 35, 42].

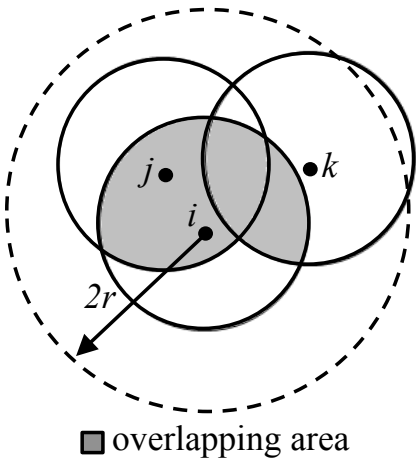


Figure 3.2: Overlapping area of sensor i with its friends j and k.

The data aggregation is usually run periodically, i.e., once in a regular interval called a round. The cluster formation in most previous approaches [13, 15, 30, 29, 17] is performed in each round. However, in our proposed protocol, the cluster formation is performed only once on the basis of the overlapping degrees of sensors at the beginning of the network operation. Each sensor tries to become a cluster head in accordance with the value of its overlapping degree i.e., the larger the overlapping degree of a sensor, the higher priority to be a head. A head with $\rho = 1$ is used only for data aggregation and transmission but not for sensing since its coverage area totally overlaps with those of its friends. Furthermore, a sensor with $\rho = 1$ other than a head is put into the sleep mode and called a *waiting node*. A waiting node does nothing but wait for the *HELP* message to replace an exhausted nearby head. If a sensor with $\rho < 1$ receives a HEAD message from one of its friends, it becomes the member of the head.

The cluster heads along with the sink are used to construct a network topology as shown in Figure 4.2, called the *backbone network*, so that each cluster head can send the aggregated data to the sink. The network topology is not changed unless any cluster-head is dying or loses the connection to the existing network. From Figure 4.2, we can see that the backbone is not a simple tree but a multi-level hierarchical network in which each node may have multiple parents belonging to the same level. Each node on the backbone can send data to the sink via its parent(s), and multiple paths may exist from a node to the sink.

3.2 Proposed Algorithms

Data are aggregated from sensors to the sink in two phases: *route construction* and *data transmission*. In the route construction phase, the sensors are grouped into clusters on the basis of their overlapping degrees, ρ_i, and then a hierarchical backbone is constructed using the cluster heads along with the sink at the top. In the data transmission phase, the sensors send their sensed data to their cluster heads and then the heads forward the data to the sink probably via multiple paths. When a head runs

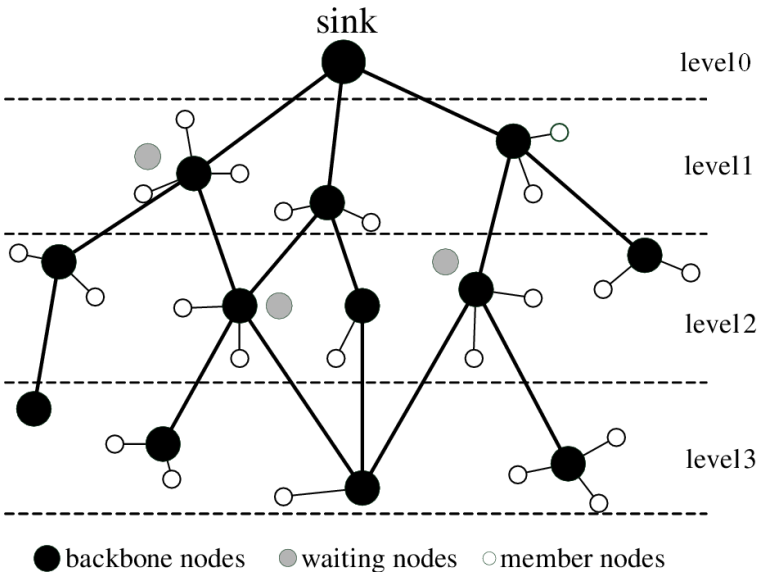


Figure 3.3: Network model.

out of its energy or its residual energy becomes lower than a predefined threshold, it drops out of the backbone and the backbone is reconfigured. For the sake of simplicity, neither the message transmission delay between the sensors nor the computation time at the sensors is taken into account.

3.2.1 Cluster Formation Algorithm

Unlike previous approaches, our proposed clustering algorithm performs the cluster formation only once at the beginning of network operation so that the overhead for clustering is greatly reduced. Furthermore, a sensor with the largest overlapping degree is selected to be the cluster head to minimize the effect of the death of the sensor. As a result, both the network lifetime and the coverage lifetime can be extended.

At the beginning, the sink broadcasts a $CLS\_FORM(T_0)$ message to inform all the sensors to start the cluster formation, where T_0 is a time limit for all the sensors to finish the cluster formation. After receiving the $CLS\_FORM(T_0)$ message, each sensor sets $D_i = (1-\rho_i)T_0$ and runs Algorithm 1 to determine its own state, i.e., head, waiting node, or member. When the timer t expires, the sensor bids for the head with its neighbors. If there is more than one sensor bidding for the head at the same time, the sensor with the smallest identification number is selected to be the head and it

broadcasts a *HEAD* message to its neighbors. If a sensor with $\rho = 1$ receives a *HEAD* message from one of its neighbors, it tries to become a waiting node. Once a sensor becomes a waiting node, it broadcasts a *SLEEP* message to its friends. If a sensor receives a *HEAD* message from a head who is one of its neighbors before its timer expires, it becomes a member of the head. On the other hand, if a sensor receives a *SLEEP* message from a waiting node, it recalculates its overlapping degree without considering the waiting nodes in its friends.

Algorithm 1 Cluster Formation Algorithm.

Initialization

```
receive CLS_FORM from the sink
find friends and neighbors
calculate ρ_i according to eq. (3.1)
set a timer t for the head determination delay D_i, and then do State Determination
```

State Determination

```
while t < D_i do
    if receive HEAD message from neighbor j then
        become member of j, and then exit
    end if
    if receive SLEEP message from friend k then
        recalculate ρ_i without considering the waiting friends
    end if
end while
if t ≥ D_i then
    bid for the head
    if ρ_i = ρ_k (i < k, k ∈ N_i) or there is no other bidding sensor then
        become the head, broadcast HEAD message to neighbors, and then exit
    else
        receive HEAD from neighbor j
        while ρ_i = 1 do
            if i < k (ρ_k = 1, k ∈ N_i) or there is no such a neighbor j then
                become waiting node, broadcast SLEEP message to friends, and then exit
            else
                receive SLEEP message from friend k and recalculate ρ_i without considering
                the waiting friends
            end if
        end while
        become member of node j and then exit
    end if
end if
```

Our cluster formation algorithm is a distributed algorithm. Each sensor, say, sensor i, exchanges its identification number, location information, and state with its

neighbors and friends. Therefore, the number of messages transferred between sensor i and its neighbors/friends is $max(O(|N_i|), O(|F_i|))$. In the worst case where all the sensors in a cluster are located in the same position, the overlapping degrees need to be recalculated for $|N_i| - 2$ times, and the computational complexity of our clustering algorithm is $O(|N_i|^2)$. Furthermore, in the worst case where all the sensors in the network are located in the same position, the computational complexity is bound by $O(|S|^2)$. However, in a general case where the sensors are well distributed in the network, we have $|N_i| \ll |S|$ and the number of clusters and the cluster size should be relatively small and the computational complexity is similar to that of previous works like LEACH [13]. Our cluster formation algorithm yields a well cluster distribution similar to the HEED but in a different sense. In HEED, a cluster head is determined on the residual energy levels of the sensors and the there should exist one cluster head within the cluster range of a sensor. On the other hand, in our proposed protocol, there exists a cluster head within the cluster range but the cluster head should be the one with the largest overlapping degree among the sensors in the cluster.

3.2.2 Backbone Construction Algorithm

The backbone is constructed by using the cluster heads along with the sink at the top. The sink initially broadcasts a *BN_CONST* message with a transmission distance of $d = R$ and with a parameter tuple $(k = 1, l = 0, id = sink_{id})$ where k denotes the parameter used for tuning the transmission distance, $d\,(= kR)$, l and id denote the level and the identification number of the message sender, respectively.

When a cluster head, e.g., node i, receives the *BN_CONST* message from its neighbors the first time, it joins the backbone and takes those neighbors as its parents, whose levels are higher than those of others. The set of parents of sensor i is denoted by P_i. After joining the backbone, node i updates its level, l, to be one lower than its parent(s). Then, node i broadcasts the *BN_CONST* message with the transmission distance d, and with a parameter tuple $(k = 1, l, id = i)$, and then sends the sink an *ON_BN* message to inform the sink of its existence on the backbone. Therefore, the sink knows whether any cluster head is still not on the backbone. If a cluster head is

not on the backbone, the sink increments the value of k and then asks the backbone nodes to broadcast the *BN_CONST* message again. The value of k is increased until the *BN_CONST* message reaches a new head that is still not on the backbone. Note that a backbone node may have multiple parents and once it connects to the backbone, its parent(s) is(are) not changed.

The backbone construction algorithm is given in Algorithm 2. The backbone construction process of a sample network using Algorithm 2 is shown in Figure 3.4. The sink initially set $k = 1, l = 0, id = sink$, and asked the nodes on the backbone to search for new heads. Node i received the *BN_CONST* message from nodes 2, 3, and 4 (Figure 3.4(a)) and then determined its parents to be nodes 2 and 3 (Figure 3.4(b)). Thereafter, node i set $k = 1, l = 2, id = i$, and broadcasted the *BN_CONST* message with $d(= R)$ but no new head could be found. The sink incremented k and asked the backbone nodes to do the search again, and node j received *BN_CONST* messages from nodes 2, 4, and i as shown in Figure 3.4(c).

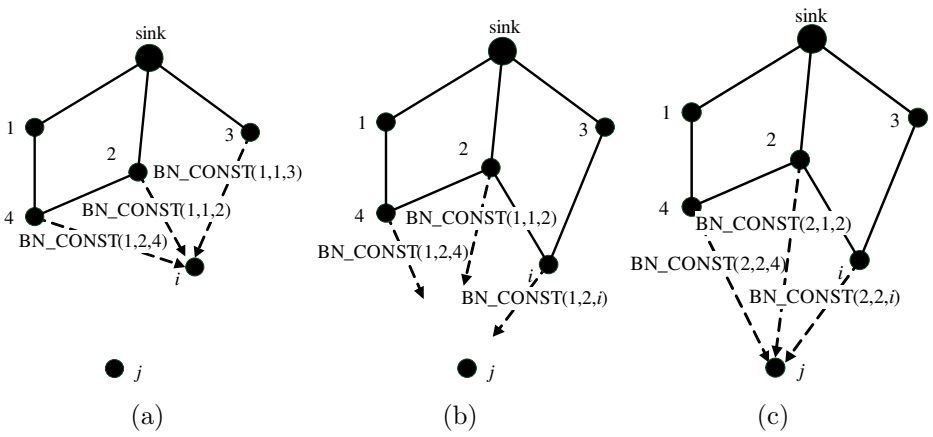


Figure 3.4: An example of a backbone construction.

In backbone construction, if the *BN_CONST* message cannot reach node i with the current value of k, the sink increments k. By assuming that the distance between node i and the sink to be d_{is}, we have $k \leq \lceil \frac{d_{is}}{R} \rceil$. For a network with high density, a node can easily find a neighbor and so k should typically be small. For example, in our simulation model with a $150 \times 150\text{m}^2$ target field, 1000 sensors, and the sink at $(150, 150)$, when we set $R = 10$, we have $k \leq 3$.

Algorithm 2 Backbone Construction Algorithm.

Procedures executed by sink

1: broadcast $BN\_CONST(k = 1, l = 0, sink_{id})$ with $d\,(= kR)$
2: **if** not receive the ON_BN message from all the cluster heads **then**
3: $k \leftarrow k + 1$ and broadcast $UPDATE(k)$ to all the backbone nodes
4: go to step 2
5: **end if**

Procedures executed by a head

1: **if** receive $BN\_CONST(k, l, id)$ at the first time **then**
2: select backbone node(s) j with the lowest level (l_{min}) as parent(s) and put it(them) in P_i
3: $l_i \leftarrow l_{min} + 1, k_i \leftarrow k$
4: send ON_BN message to sink
5: broadcast BN_CONST with a tuple (k, l_i, i) and distance $d\,(= kR)$
6: **end if**
7: **if** already on backbone **and** receive $UPDATE(k)$ from sink **then**
8: broadcast BN_CONST with a tuple (k, l_i, i) and distance $d\,(= kR)$
9: **end if**

3.2.3 Flow-Balanced Routing Algorithm

Each backbone node, i.e., cluster head, collects the sensed data from its members and then conveys the collected data to the sink. A backbone node may have multiple parents and therefore may have multiple paths to the sink as shown in Figure 3.5. Our goal is to balance the residual energy of each backbone node in order to prolong the network lifetime, that is, to equalize the residual energy levels of the parents of a sensor after sending the collected data. For example, assuming that a backbone node, say, node i, had I-bit data to the sink and three parents whose current energy magnitudes were 0.1J, 0.3J, and 0.4J, respectively. After transferring the data, the residual energy magnitudes of the parents were equalized; e.g., they would become 0.1J, 0.2J, and 0.2J, respectively.

The energy needed to send one bit data from one sensor to another can be calculated by using the $1/s^n$ path loss model [43] as follows.

$$P_{relay}(s) = (\alpha_1 + \alpha_2 s^n)\gamma, \tag{3.2}$$

where s is the transmission distance, α_1 is the total energy per bit consumed by the transmitter and the receiver electronics, α_2 accounts for energy dissipated in the

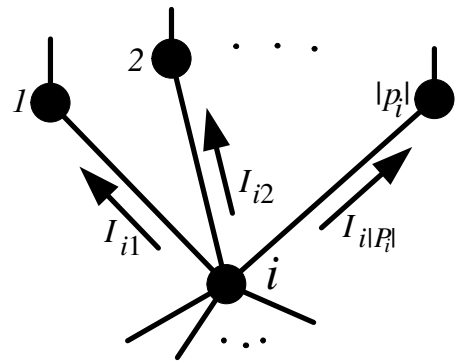


Figure 3.5: Multiple paths from node i to the sink.

transmit op-amp, γ is the number of bits relayed per second, n is the path loss exponent, and typically n takes a value between 2 and 5. Similar to Heizelman et al. [13], we calculate the transmission energy by using the free space (s^2 power loss) and the multi-path fading (s^4 power loss) channel models as follows.

$$E_t = \begin{cases} (E_e + \epsilon f s^2)I, & \text{if } s < s_0, \\ (E_e + \epsilon m s^4)I, & \text{if } s \geq s_0, \end{cases} \tag{3.3}$$

where $s_0 = \sqrt{\frac{\epsilon f}{\epsilon m}}$, and E_e is equivalent to α_1 in (2) and we set it to 50nJ/bit. Since the power control can be used to invert this loss by appropriately setting the power amplifier, if the distance is less than a threshold s_0, the free space (fs) model is used, that is, $n = 2$ and α_2 is set to $\epsilon f = 10pJ/bit/m^2$. Otherwise, the multipath (mp) model is used, that is, $n = 4$ and α_2 is set to $\epsilon m = 0.0013pJ/bit/m^4$. The energy needed to receive I-bit data can be calculated by

$$E_r = E_e I. \tag{3.4}$$

Since each backbone node may have multiple parents, the energy needed to send a given size message to each of its parents needs to be estimated. Let E_j ($j \in P_i$) to denote the residual energy of node i's parent j. Assuming that node i has I-bit data, denoted by I_i, to the sink and that the flow from node i to its parent j is denoted by

I_{ij}, then we have

$$I_i = \sum_{j \in P_i} I_{ij}. \tag{3.5}$$

Therefore, we can write the energy/bit consumption for conveying data I_{ij} at node j as

$$\Delta E_{ij} = (\alpha_1 + \alpha_2 (k_j R)^n) I_{ij} = \varepsilon_j I_{ij}, \tag{3.6}$$

where $\varepsilon_j = (\alpha_1 + \alpha_2 (k_j R)^n)$. Therefore, the residual energy of node i's parent j, denoted by X_j, after conveying data I_{ij} can be written as

$$X_j = E_j - \Delta E_{ij} = E_j - \varepsilon_j I_{ij}. \tag{3.7}$$

Here, we attempt to let the following relation hold.

$$X_j = \bar{E} = \frac{1}{|P_i'|} \sum_{l \in P_i'} X_j, \;\; j \in P_i', \tag{3.8}$$

where P_i' ($P_i' \subseteq P_i$) denotes the set of node i's parents to which the flow from node i is greater than 0, i.e., for $E_j > \bar{E}$ ($j \in P_i'$), $I_{ij} > 0$. According to Eqs. (3.5), (3.6), (3.7), and (3.8), we can determine I_{ij}. On the other hand, for $E_j \leq \bar{E}$ ($j \in P_i \setminus P_i'$), the flow from node i to node j should be 0, i.e., $I_{ij} = 0$. Therefore, we have for $j \in P_i$

$$I_{ij} = \begin{cases} 0, & \text{if } E_j \leq \bar{E}, \\ \dfrac{I_i + E_j \sum\limits_{l \in P_i'} \dfrac{1}{\varepsilon_l} - \sum\limits_{l \in P_i'} \dfrac{E_l}{\varepsilon_l}}{\varepsilon_j \sum\limits_{l \in P_i'} \dfrac{1}{\varepsilon_l}}, & \text{if } E_j > \bar{E}. \end{cases} \tag{3.9}$$

The proposed routing algorithm is executed by each node to determine the flow to each of its parents that satisfies Eq. (3.9).

In flow-balanced routing, attempts are made to equalize the residual energy of node i's parents after data transmission. The calculation of I_{ij} (line 3 in Algorithm 3) plays a key role in determining the capable parents to which node i can send some data. First, node i calculates the total energy needed to convey data I_i and estimates

Algorithm 3 Data Aggregation Algorithm.

1: get residual energy of parents $E_j (j \in P_i)$
2: $P'_i \leftarrow P_i$
3: calculate I_{ij}
4: **if** there is any parent $j, I_{ij} < 0$ **then**
5: set $I_{ij} = 0$ and remove j from P'_i
6: goto step 3
7: **end if**
8: send flow I_{ij} to parent j $(j \in P_i)$ that satisfy eq. (3.9)

the average energy of its capable parents by ignoring those parents with energy lower than the average. This process is repeated until all the capable parents have energy equal to or more than the average energy. The computational complexity of this process is bound by $O(|P_i|^2)$. Note that only the total residual energy of node i's parents are considered here, and if node i's parents do not have enough energy to convey the collected data, node i is regarded as an isolated node with no capable parent. Then, the isolated node reconnection mechanism described in Section 3.2.4 is triggered.

3.2.4 Rerouting Algorithm

Instead of reconstructing the whole backbone in each round, we propose a local rerouting algorithm to do the backbone reconfiguration only if the topological change occurs in any place; i.e., if a node drops out of the backbone due to its energy exhaustion. If the residual energy of a backbone node becomes lower than a predefined threshold, e.g., a given percentage, denoted by r_{th}, of sensor's initial energy, the node's energy is exhausted and the rerouting algorithm is triggered. The exhausted head tries to find a new head in its neighbors to replace itself. Otherwise, its descendants including the members in its cluster and the children on the backbone lose the connection to the sink, and have to try to repair the connection to the network by themselves. Our proposed rerouting algorithm contains two phases: *head replacement* and *isolated node reconnection*.

In the *head replacement* phase, the exhausted head, e.g., node i, broadcasts a *HELP* message to its neighbors to find a capable node to replace itself. A waiting

Algorithm 4 Rerouting Algorithm.

Head Replacement executed by exhausted head i

broadcast *HELP* message to neighbors
if receive response(s) from neighbor(s) **then**
  select the best node, denoted by j, as new head
  if node j is not a backbone node **then**
    send $REQ\_HD(k_i, l_i, i)$ message to node j
  end if
  become a member of node j, and inform members and children to connect to node j
else
  ask members and children do **Reconnection**
end if

Reconnection executed by an isolated member i

if find a backbone node j in neighbors **then**
  become member of node j
else
  if find a node, j $(j = \arg\max(E_k), k \in N_i, E_k/E_0 > r_{th}, E_k > E_i)$ **then**
    send $REQ\_HD(k_i, l_i, i)$ message to node j, and become member of node j
  else
    become new head and broadcast $RECON(k_i, l_i, i)$ within $d\,(= k_i R)$
    while cannot find a backbone node l within $d\,(= k_i R)$ whose level is higher than l_i **do**
      $k_i = k_i + 1$
    end while
    become child of node l
  end if
end if

Reconnection executed by an isolated head j

broadcast $RECON(k_j, l_i, j)$ within $d\,(= k_j R)$
while cannot find a backbone node l within $d\,(= k_j R)$ whose level is higher than l_j **do**
  $k_j = k_j + 1$
end while
become a child of node l

node has a higher priority to become the new head. To detect the HELP message, the waiting node may keep the radio receiver on in each round. Considering that overhearing is not energy efficient [44], we can borrow the beacon scheduling way from [45] by setting the radio receiver on only at the beginning of each round to save energy. In this thesis, we also propose a new scheme for the waiting node to detect HELP message. When the data aggregation begins, a waiting node checks the residual energy of each head in its neighbors at the beginning of the first and the second rounds to estimate the remaining lifetime of the head, denoted by t_r, i.e., $t_r \approx \lfloor \frac{E_2 - E_1 r_{th}}{E_1 - E_2} \rfloor$ where E_1 and E_2 denote the residual energy at the beginnings of the first and the second rounds, respectively. To prevent a waiting node from oversleeping, we can use a parameter $\beta\,(0 < \beta < 1)$ and determine t_r as follows.

$$t_r = \lfloor \beta \frac{E_2 - E_1 r_{th}}{E_1 - E_2} \rfloor. \tag{3.10}$$

Since there may be more than one waiting node in the exhausted node's neighbors, the waiting node with the smallest *id* number is selected as the new head. However, if no waiting node can be found, the node that has the most residual energy and the ratio of the residual energy to the initial energy is higher than r_{th} is selected as the new head. After the new head has been determined, the exhausted head, say, node i, informs its members and children of the result. If the new head, say, node j, is not a backbone node, node i sends a $REQ\_HD(k_i, l_i, i)$ message to j so that j can determine its level and data transmission distance. In the worst case, if node i cannot find any candidate node to replace itself, it involuntarily throws away its descendant(s), and just transfers its own sensed data to its parent node(s) until its death. The abandoned members and children become isolated orphans, and they have to find their new head or parent(s) by themselves.

In the *isolated node reconnection* phase, a member of the exhausted head who realizes its cluster head has gone will try to find a new head. If it can successfully find a backbone node in its neighbors, it becomes its member immediately. However, if it cannot find any backbone node, but has more energy than its neighbors, it becomes the new head and sets $k = 1$. Otherwise, it invites the neighbor with the most energy

to become the new head, say, node j, and set $k_j = 1$.

An isolated head, say, node j, broadcasts a $RECON(k_j, l_j, j)$ message to connect to the backbone. If no backbone node responds, it increments k_j and then broadcasts the $RECON$ message again until it finds a backbone node. The rerouting algorithm is illustrated in Algorithm 4.

3.3 Performance Evaluation

3.3.1 Simulation Model

We compared the performance of our proposed FBR with those of LEACH, HEED, M-HEED, CPCP, and HEED-FBR using simulation. Two performance metrics, *network lifetime* and *coverage lifetime*, are used for comparison. As described in section 2.2, the network lifetime is defined as the number of data aggregation rounds from the beginning instant of the network operation to the instant when any or a given percentage of the sensors die. On the other hand, the coverage lifetime is defined as the number of data aggregation rounds from the beginning instant of the network operation to the instant when the ratio of the coverage of the current alive sensors to the coverage of the whole sensors drops below a predefined threshold. The M-HEED approach is a multi-hop hierarchical version of HEED developed in this thesis wherein the cluster heads are constructed hierarchically using the recursive approach proposed in HEED. Furthermore, HEED-FBR is a modified version of FBR wherein the cluster formation algorithm is replaced by HEED.

Our simulation program was developed using Java. The parameter values used in the simulation experiments are shown in Table 4.1, and the performance metrics were also examined with various parameter values. A network model with a 150×150 m^2 square area was used. The sensors were randomly deployed in the network but the sink was located at the position (150, 150). All sensors had an initial energy of $0.5J$. We assumed that each sensor was assigned a unique identification number. In practice, a method [24] can be used to assign distinct identification numbers to the sensors. The data from each sensor to the sink were assumed to be 2000 bits, and the

Table 3.1: Parameters used in simulation

Parameter name	Symbol	Value
Field size		150×150 m^2
Sink location		(150,150)
Number of sensors	S	1000
Sensing range	r	5m
Cluster range	R	10m
Transmission tuning parameter	k	≥ 1
Initial energy	E_0	0.5J
Transmission energy/bit	E_e	50nJ/bit
Amplifier energy(fs)	ϵf	10pJ/bit/m^2
Amplifier energy(mp)	ϵm	0.0013pJ/bit/m^4
Data size	I	2000bits
Control message size	msg	100 bits
Data compression ratio	r_{dc}	30%
Energy threshold ratio	r_{th}	30%
Head percentage (LEACH)	p	5%
Head percentage (HEED)	C_{prob}	5%
Energy threshold (HEED,CPCP)	P_{min}	10^{-4}J
Transmission range(broadcast) (CPCP)	R_{bc}	20m

sizes of the control messages exchanged between sensors and between a sensor and the sink were assumed to be all the same and were 100 bits. Time in the experiments was proceeded in rounds similar to previous protocols [13, 15]. At the beginning of each round, the cluster formation is performed in previous protocols but in our proposed protocol it is performed only once at the beginning of the network operation. To avoid any unfair treatment over previous approaches, we simulated those protocols with a wide range of parameter settings and chose the best parameter combinations for the comparison. Each simulation experiment has been repeated 10 times in order to calculate the confidence intervals of the results.

3.3.2 Performance Comparison

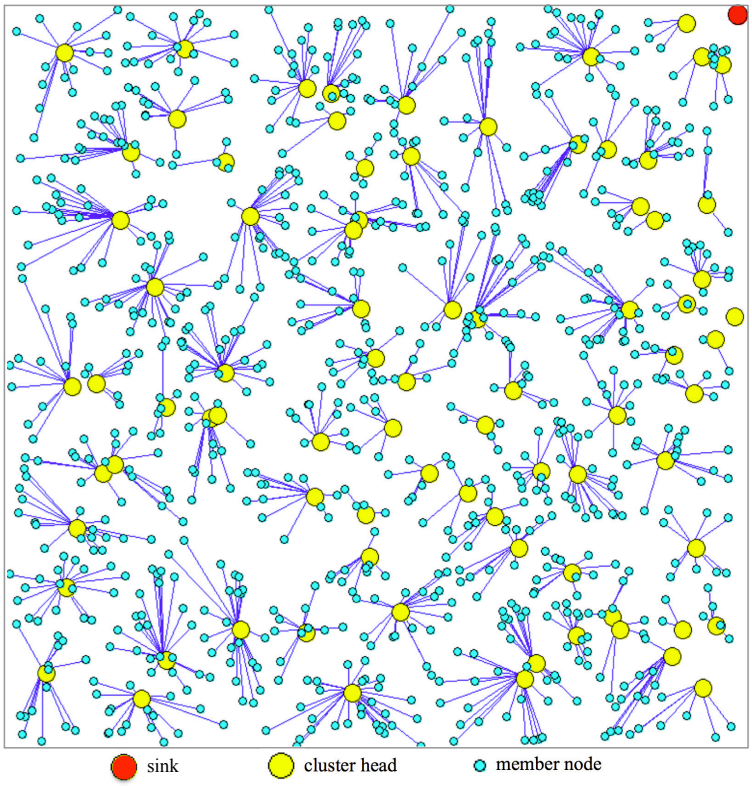
Examples of network topologies constructed using previous approaches are shown in Figure 3.6 and Figure 3.7. The sink colored in red was placed in the top-right corner and sensors are randomly placed in target filed, circles colored in cyan, yellow, and gray denote member nodes, cluster heads, and sleep nodes, respectively. From Figure 3.6(a), we see that the cluster heads distributed randomly as each sensor can elect to be a head with a certain probability (5%). In contrast, HEED shown in 3.6(b)

obtained a well cluster head distribution as the head election mechanism rules sensors with higher residual energy and lower communication cost prior to becoming cluster head within a certain cluster range.

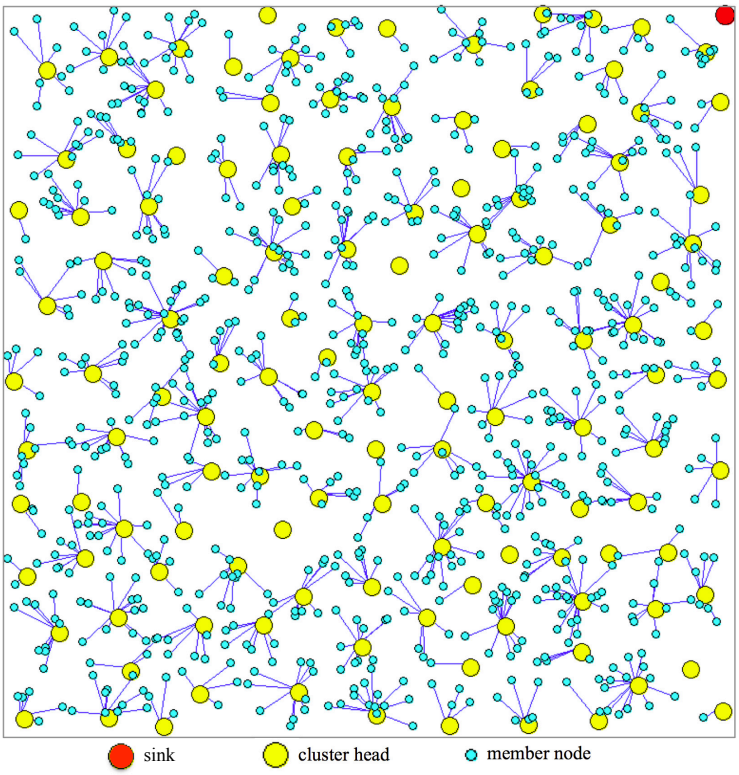
Figure 3.7(a) illustrates an example of network constructed by recursively executing HEED approach, where the cluster range get a one-time increase when HEED algorithm is to be executed in the next level. For example, initially, we set the cluster range to be 10 and determine some cluster heads, colored in yellow in Figure 3.7(a), from all sensors, then we set the cluster range to be 20 and selecte the next level cluster heads, colored in pink, from the previous level cluster heads colored in yellow. M-HEED obtained a multi-hop hierarchical network and the cluster head distributed well in each level. However, when the sink is located at the side of target area (not at the center), more generally, data generated by sensors closer to the sink may be conveyed to a upper layer head further to the sink. The example network topology constructed by CPCP is shown in Figure 3.7(b), some sensors colored in gray turn into sleep model, and the active nodes construct a tree with the leaves of cluster members colored in cyan and non-leaves of cluster heads colored in yellow.

One example of the network topology obtained using our FBR protocol is shown in Figure 3.8. From this figure, we see that the backbone topology is a novel multi-level structure, rather than a simple tree, one node may have multiple parents in the same level, and there are multiple paths from the node to the sink.

Furthermore, in the direction of a path from a sensor to the sink, the workload of each node gradually increase due to the data aggregation level by level. On the other hand, in our backbone construction algorithm, backbone is constructed from the sink by broadcasting message within a predetermined distance, this distance increases when some new heads can not connect to the network. As a result, the communication distance between two levels from a sensor to the sink exhibits a non-increasing trend, this would decrease the negative effects generated by the gradually increased workload. However, it is more attractive to search optimal values of level distances, this is also for other factors such as the number of sensors, cluster range, and so on. Optimal problems for wireless sensor networks will be considered as other matters in our future

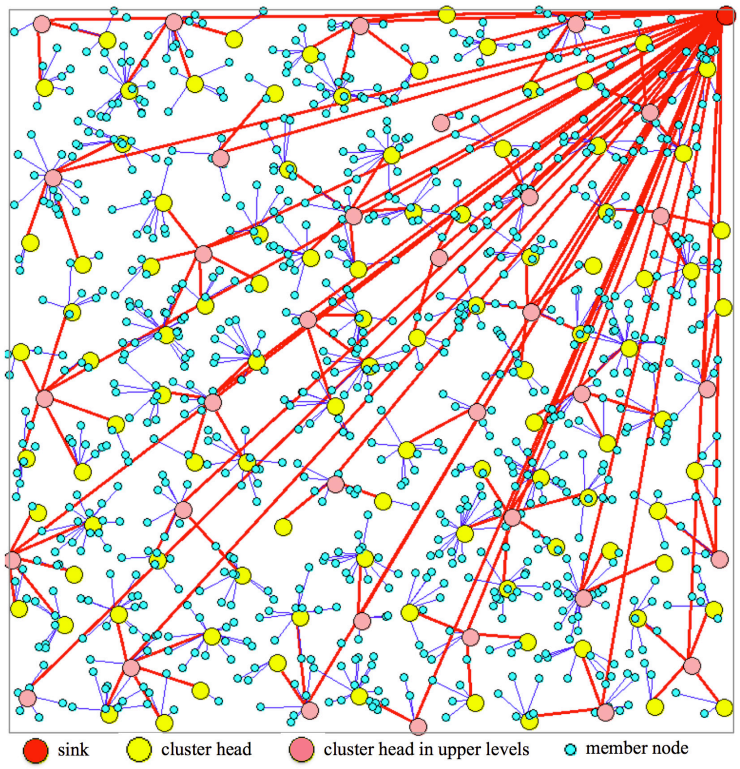


(a) LEACH

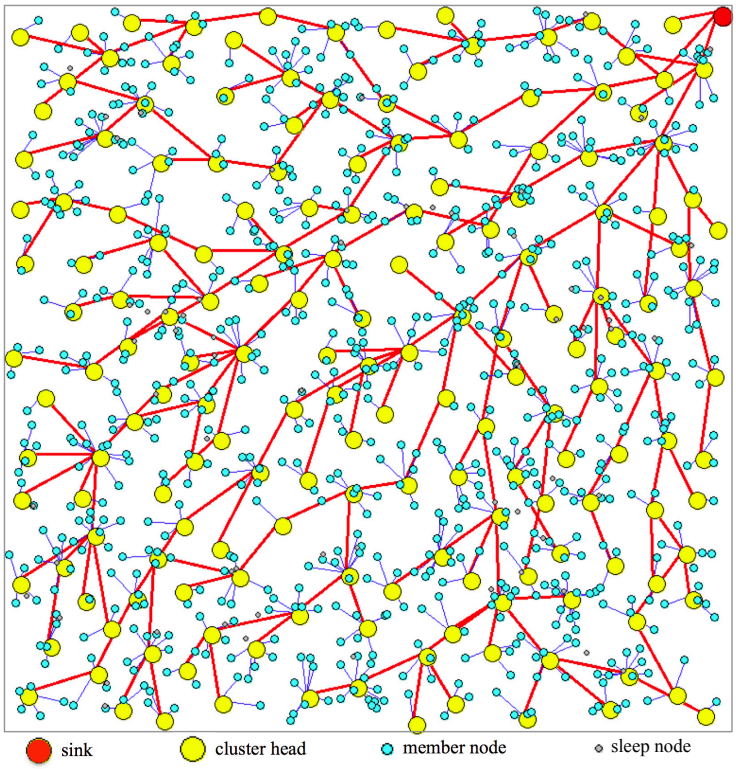


(b) HEED

Figure 3.6: Examples of network topologies constructed in single cluster approaches.



(a) M-HEED



(b) CPCP

Figure 3.7: Examples of network topologies constructed in multi-hop approaches.

work.

Another example of network topology constructed using HEED-FBR, a modified version of FBR wherein the cluster formation algorithm is replaced by HEED, is shown in Figure 3.9.

Figure 3.10 compares the network lifetime of our FBR protocol with those of LEACH, HEED, M-HEED, CPCP, and HEED-FBR. The horizontal axis means the number of rounds in which sensors complete data aggregation one time, and the vertical axis means the percentage of alive sensors.The results shown in the figure were obtained as the sample means of 10 experiments with 95% confidence intervals. Since the half widths of the confidence intervals are all less than 2% of the sample means, they are not shown in the figures. We see from this figure that FBR yields a much longer lifetime than the others. The lifetime of FBR when the first sensor died is near 10 times longer than that of the best pervious work CPCP and is around 5 times longer when 10 percent of the sensors have died. Furthermore, we can see that HEED-FBR also provides a long lifetime. The difference between the lifetimes of FBR and HEED-FBR shows the usability of the proposed cluster formation approach. Similarly, by comparing HEED-FBR and HEED, we see that the flow-balanced routing algorithm plays a key role in data aggregation and that balancing flows between nodes yields a long lifetime.

In FBR, flow-balanced routing algorithm is executed by a node with information of multiple parents, residual energy of nodes in the same level can be nearly equalized. At the same time, data flow dispersion over the network is realized. To some extent, energy of all nodes is tend to be equalized. This can be read from the results of experiments shown in Figure 3.10, where all nodes obtained a nearly equalized lifetime in FBR.

Figure 3.11 shows the coverage lifetimes of FBR along with HEED, CPCP, and HEED-FBR. We selected these previous protocols for comparison because both HEED and CPCP are coverage-aware protocols [17]. From this figure, we see that the coverage lifetime of CPCP largely outperforms HEED in contrast to the network lifetime shown in Figure 3.10. Furthermore, both FBR and HEED-FBR yield much longer

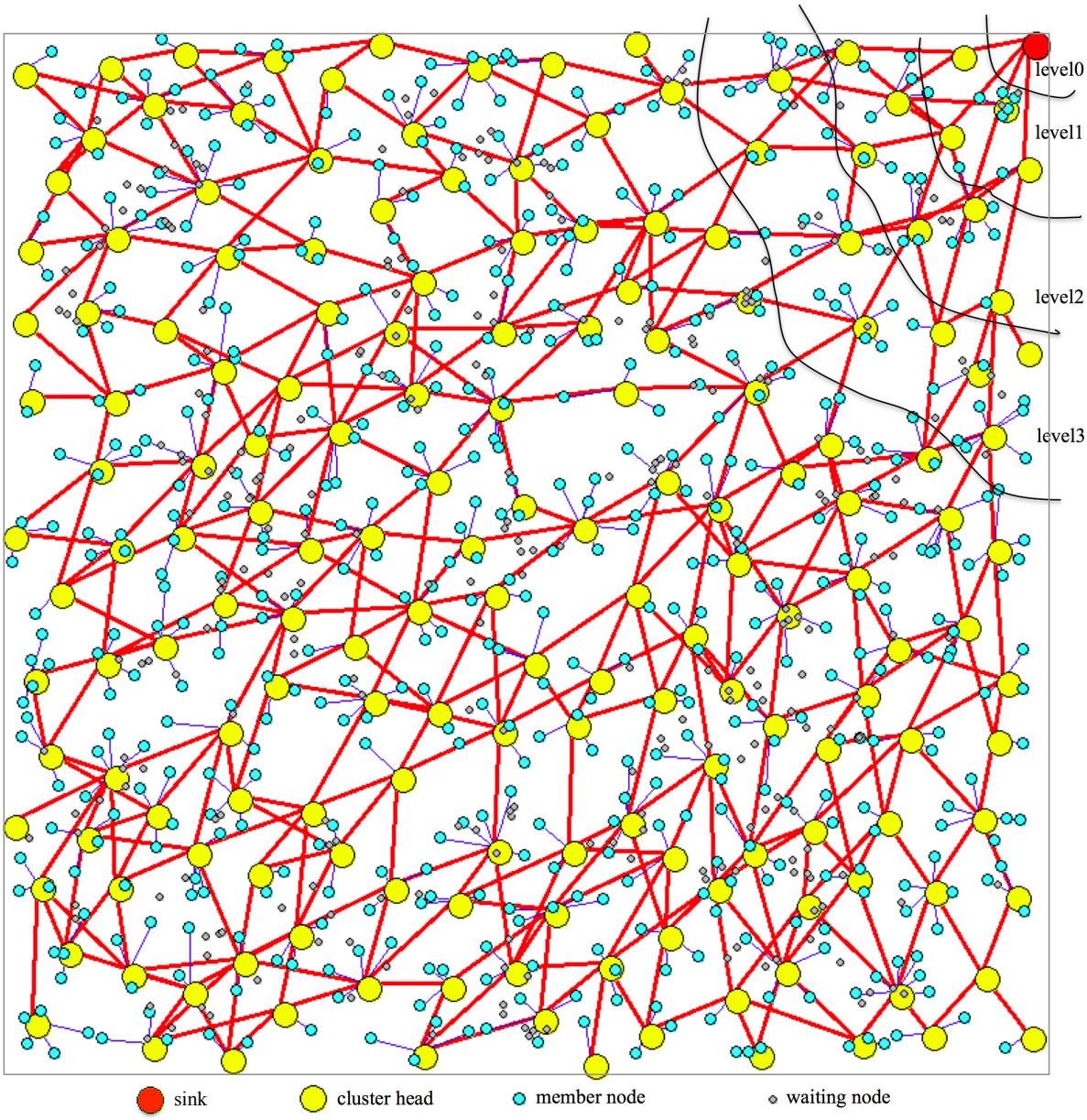


Figure 3.8: An example of the hierarchical network topology constructed in FBR.

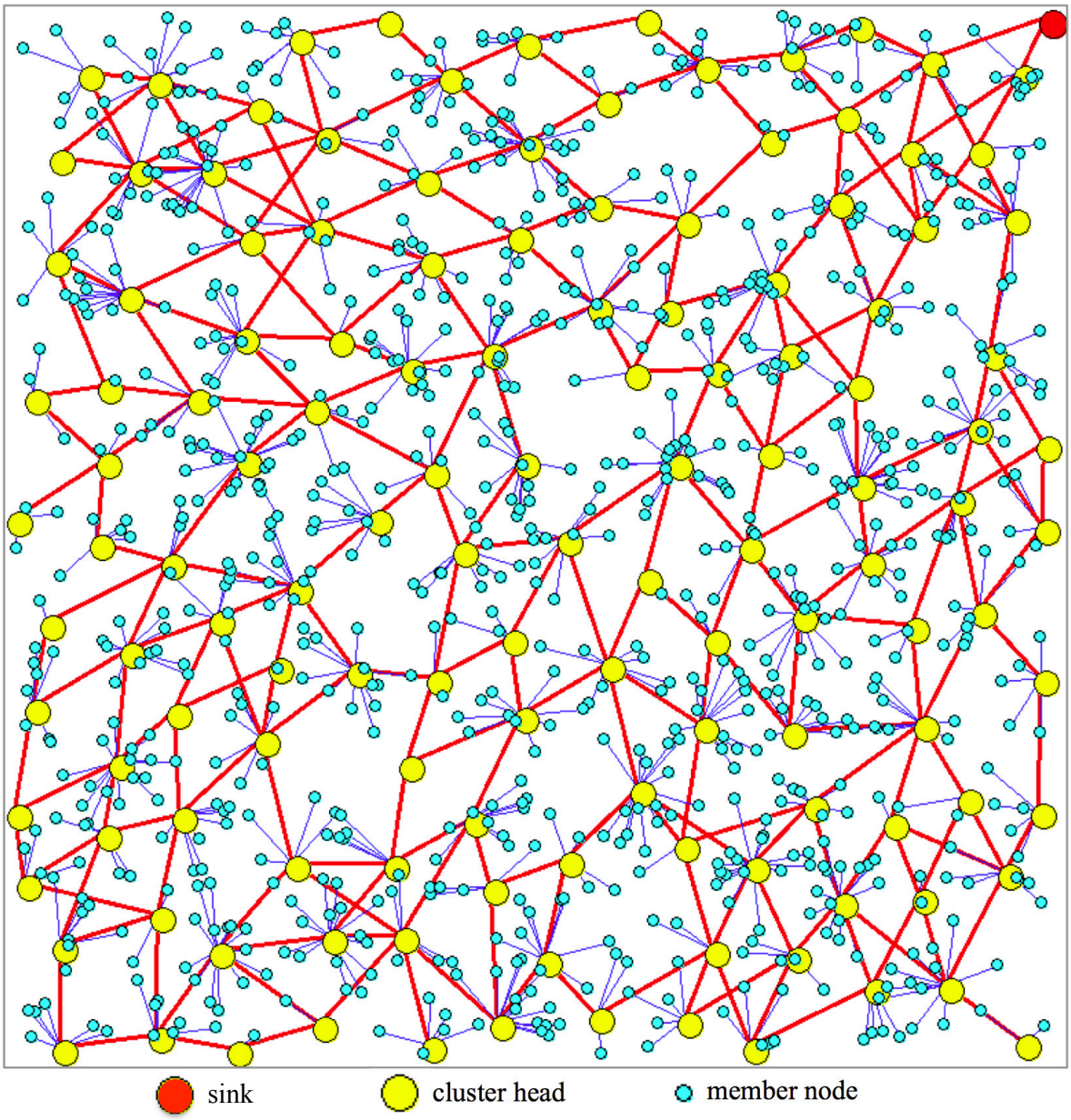


Figure 3.9: An example of the hierarchical network topology constructed in HEED-FBR.

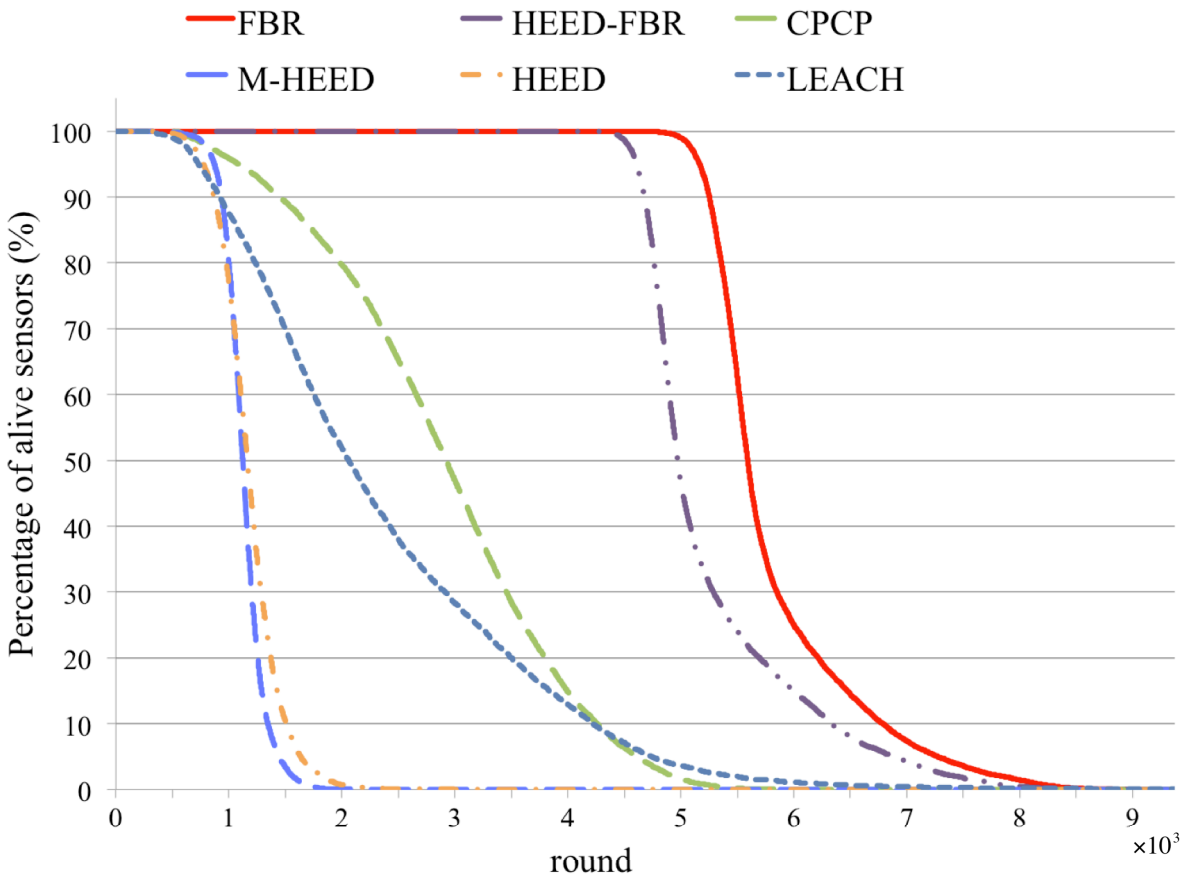


Figure 3.10: Network lifetimes of various protocols.

coverage lifetimes than HEED and CPCP.

The main reasons for the above results can be summarized as follows.

1) The backbone constructed in FBR is not a simple tree but a multi-level structure with the sink at the top. Each head may have multiple paths to the sink and therefore balancing the flow from a sensor to the sink over multiple paths can equalize the energy consumption among the heads, resulting in a longer lifetime. On the other hand, in the multi-level protocols extended on the basis of HEED, attempts are made for the cluster heads at each level to be distributed uniformly in the network, resulting in some higher level heads far away from the sink. Those heads have to spend more energy to send data to the sink and die fast. Furthermore, in CPCP the cluster heads are simply constructed as a shortest path tree rooted at the sink. A head near to the sink and with more offspring should die quickly.

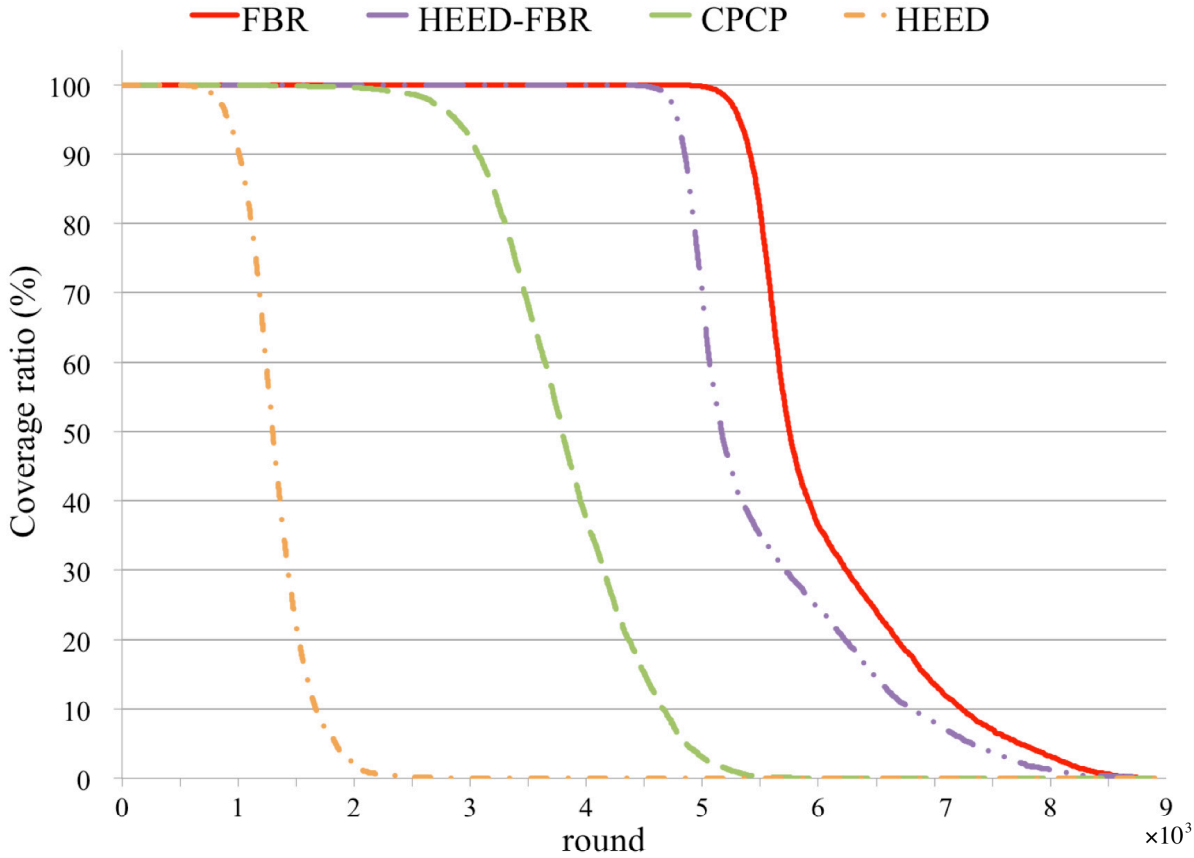


Figure 3.11: Coverage lifetimes of various protocols.

2) A local rerouting approach is used in FBR (and HEED-FBR) to repair the backbone topologyonly at the location where the topological changes occur. In the previous algorithms, on the other hand, the network construction is repeatedly executed at the beginning of each round.

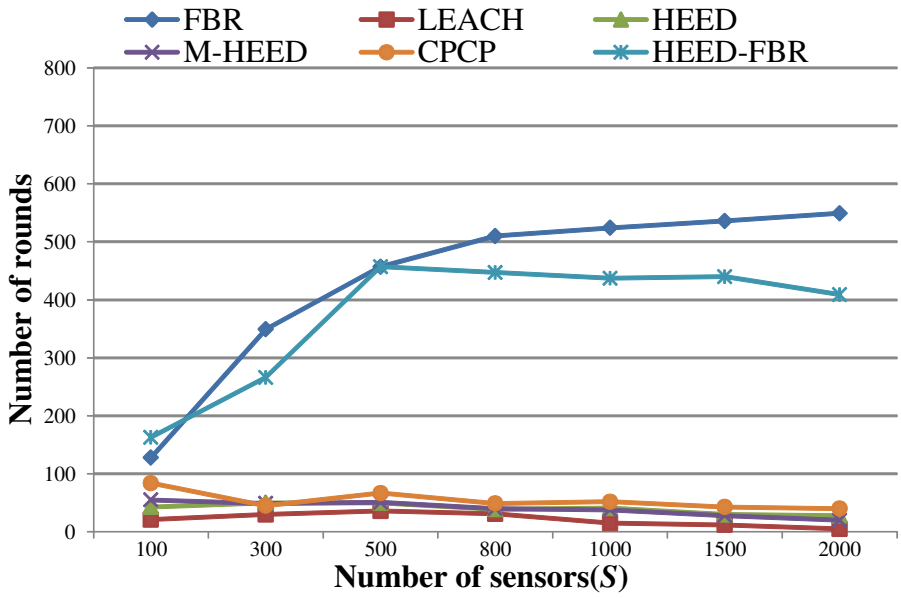
3) In large-scale sensor networks, a large number of sensors are usually deployed randomly in the target field. The coverage areas of some sensors may totally overlap those of others. Taking out the overlapped sensors does not degrade the usability of the network at all. In FBR, the overlapping sensors are taken as *waiting nodes*, that is, those sensors are put into the sleep mode to reduce energy consumption.

3.3.3 Effects of System Parameters

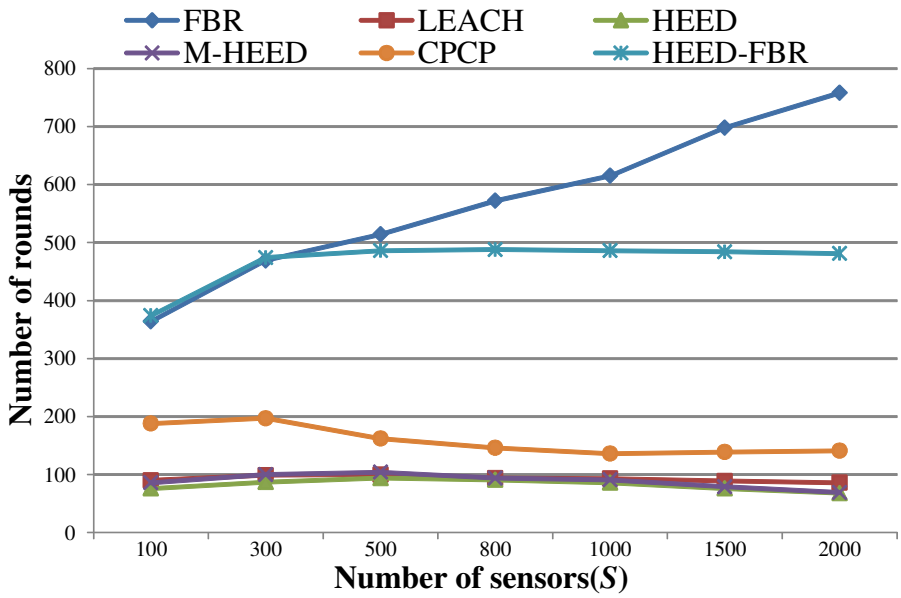
To further examine the effects of the system parameters on the performance of our proposed protocol, we simulated FBR and also the main previous protocols with various parameter settings as follows. In these experiments, the parameter being examined is changed while all others are fixed. The initial energy of each sensor was set to 0.05J in order to speed up the experiments, and the others are shown in Table 4.1. For the sake of simplicity, the figures show only the network lifetimes of the protocols under consideration when the first sensor or ten percent of the sensors died.

- The numbers of sensors were set to 100, 300, 500, 800, 1000, 1500, and 2000.
- The values of sensing range r were set to 2, 5, 8, 10, and 15.
- The cluster ranges R were set to 5, 10, 15, and 20.
- The energy threshold ratios r_{th} were set to 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.8, and 0.9. This indicates the ratio of the current residual energy of a head to its initial energy and works similarly to parameter p_{min} in HEED and CPCP, but is different in the sense that r_{th} can be adjusted to adapt to various conditions.
- The data compression ratios r_{dc} were set to 0.2, 0.3, 0.4, 0.5, and 0.6. This indicates the ratio of the size of the aggregated data at a head to the total size of the original data received from its members along with its own sensed data. If a head receives nI-bit data from its children and generates I-bit data to send, then the total data sent to the sink are $r_{dc}(n+1)I$ bits. In the experiments, we also simulated a special case, similar to LEACH, HEED, and CPCP, wherein the data collected at a cluster head are aggregated into one packet no matter how many packets the head receives. The result of this case is shown by η in Figure 3.16.

Figure 3.12 illustrates the lifetimes of the algorithms under consideration when changing the number of sensors. We see that when the number of sensors increases



(a) Lifetime when the first sensor died.



(b) Lifetime when 10% sensors died.

Figure 3.12: Lifetimes for various network sizes.

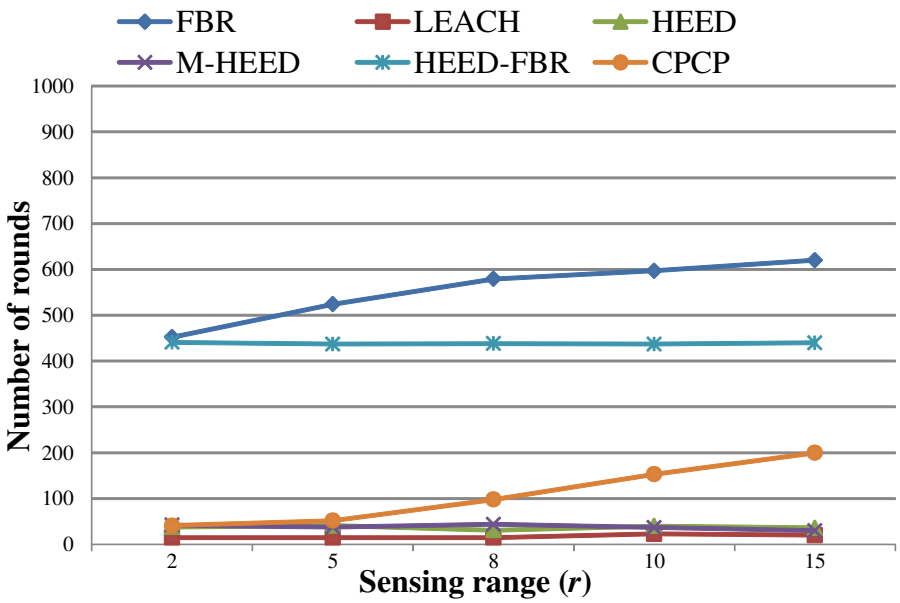
FBR outperforms others, because more nodes are treated as waiting nodes and the number of paths from each node to the sink may increase, resulting in better flow balancing. Even though CPCP puts some sensors into sleep mode whose sensing areas are totally covered by other sensors, this is decided after the cluster formation phase, but there is no need to consider the sleep nodes in cluster formation. Furthermore, when the number of sensors increases, the network construction overhead in each round also increases, wasting scarce resources, so rerouting locally would obviously work more efficiently than reconstructing the network.

Figure 3.13 shows the lifetimes of the algorithms for various sensing ranges. We see that FBR outperforms other protocols as the sensing range increases, because when the coverage area of each sensor increases, the number of nodes that can become waiting nodes also increases. Furthermore, a sensor can find more friends in its widened sensing area and may also choose a better head.

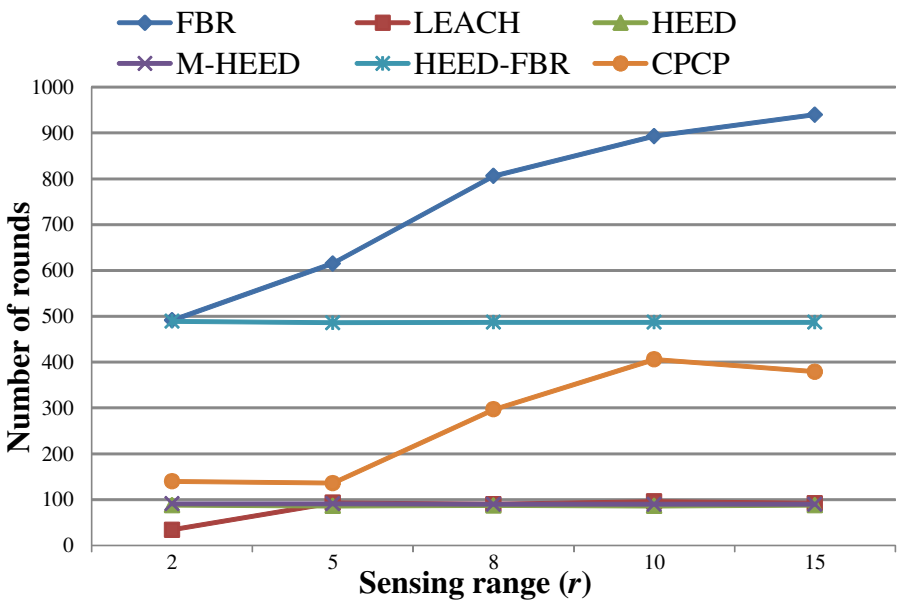
Figure 3.14 shows the lifetimes of the algorithms for different cluster ranges. We see that the cluster range, also used in HEED, M-HEED, and CPCP, affects the performance more because the cluster range determines the size of clusters and the number of network levels.

Figure 3.15 shows the lifetimes for different head energy threshold ratios, we find that the r_{th} is a sensitive parameter for FBR, while in other algorithms the effect of this parameter can be negligible. From Figure 3.15, we see that FBR performs best when r_{th} is around 0.25-0.4. Since r_{th} is a tunable parameter, we can adjust its value depending on the network configuration.

Figure 3.16 shows the lifetimes of the algorithms when changing data compression ratios. In this figure, η denotes an extreme case: no matter how many data packets a node receives it will aggregate them into one packet. We see from this figure that FBR outperforms the others and that the network lifetime decreases when the compression ratio becomes low.

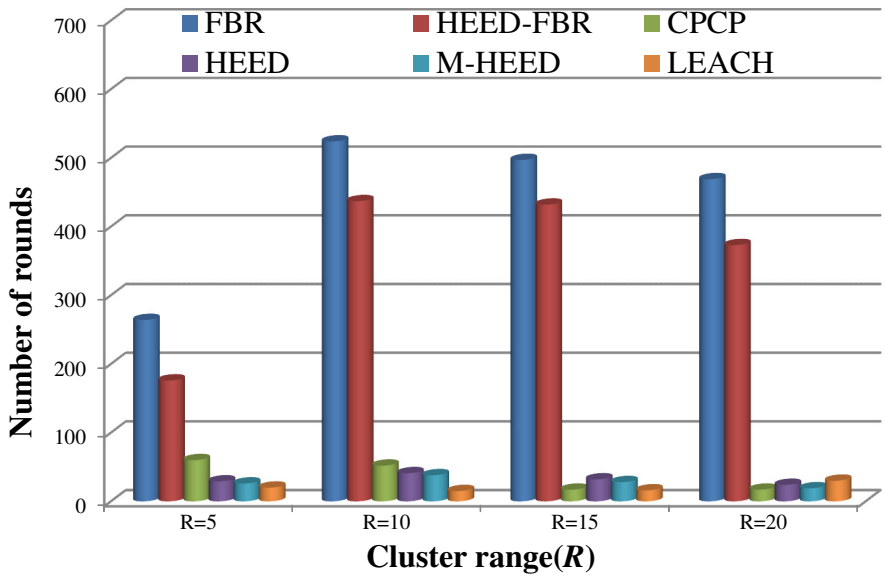


(a) Network lifetime when the first sensor died.

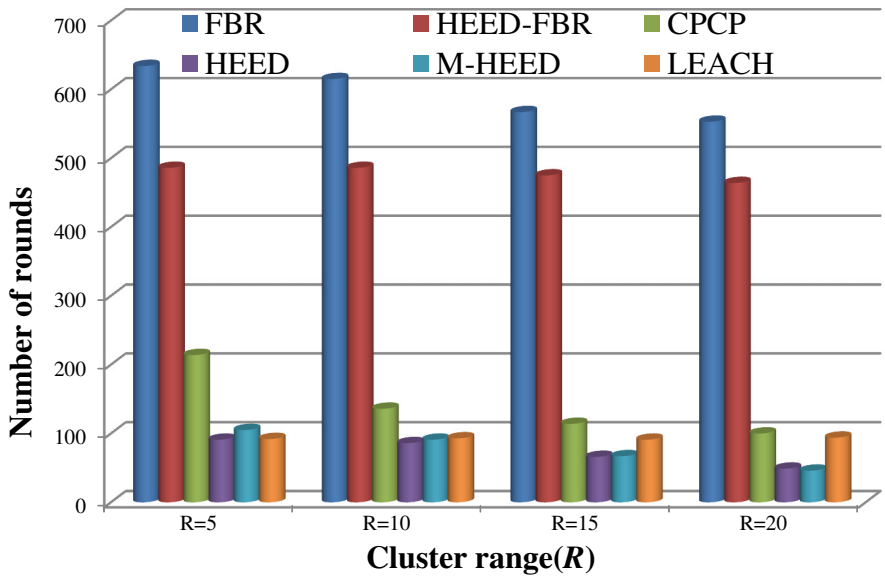


(b) Network lifetime when 10% sensors died.

Figure 3.13: Lifetimes for various sensing ranges.

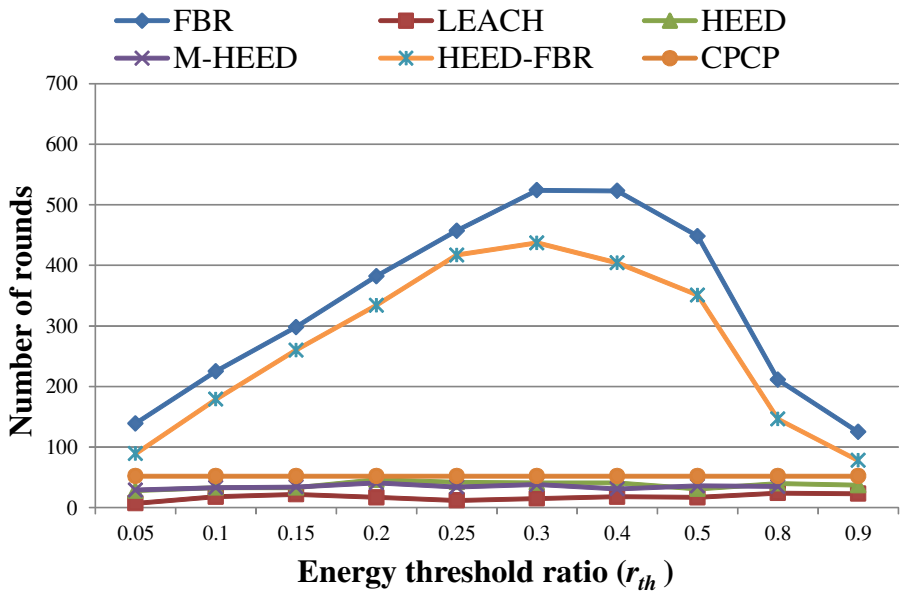


(a) Network lifetime when the first sensor died.

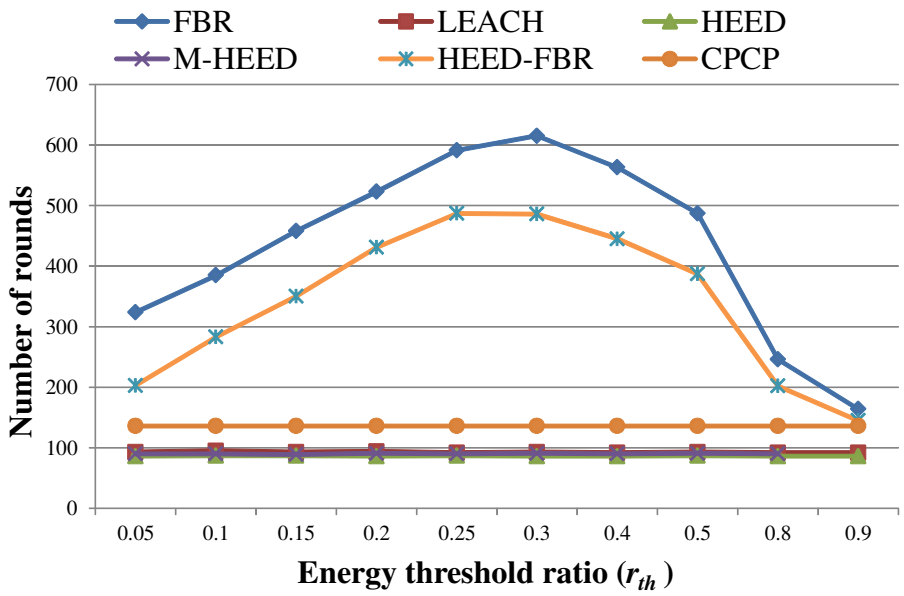


(b) Network lifetime when 10% sensors died.

Figure 3.14: Lifetimes for various cluster ranges.

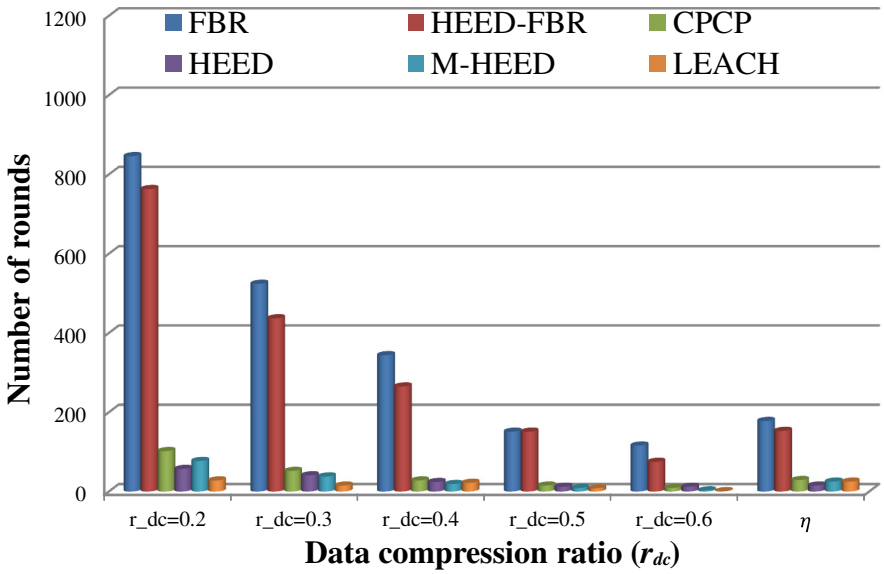


(a) Network lifetime when the first sensor died.

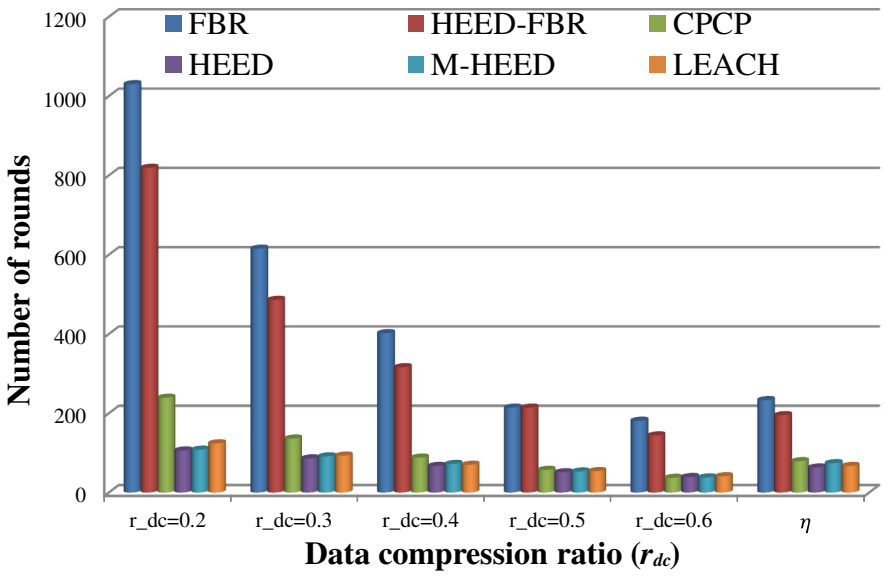


(b) Network lifetime when 10% sensors died.

Figure 3.15: Lifetimes for various head energy thresholds.



(a) Network lifetime when the first sensor died.



(b) Network lifetime when10% sensors died.

Figure 3.16: Lifetimes for various data compression ratios.

3.3.4 Performance Examination of An Extreme Case

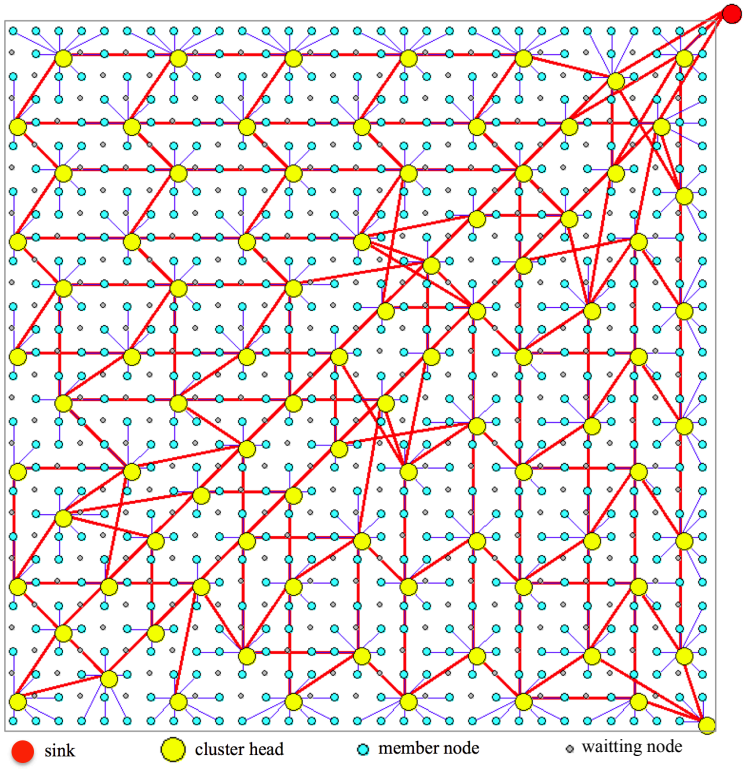
In order to display the characteristics of network topologies constructed by our protocol more intuitively. We also do simulation under an extreme case where 961 sensors are pre-deployed in a lattice with 31×31. The other parameter settings are the same as above shown in Table 4.1. The network topologies constructed by FBR and CPCP are given in Figure 3.17. From these two topologies, we can see that our FBR results a better balance than CPCP. The corresponding network lifetimes when the first node dies, 10% of sensors die, and 30% of sensors die are given in Table 3.2. From these data, we can see that FBR also performs better than CPCP in this case. When the sensors are deployed regularly, the good performance of FBR is more significant.

Table 3.2: Network lifetime when sensors are deployed in lattice.

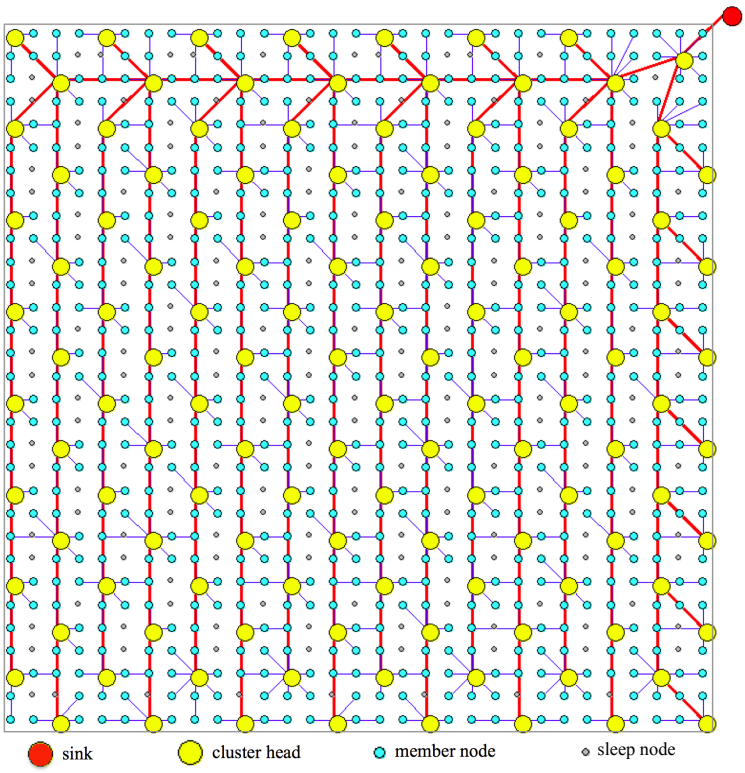
	FBR	CPCP
first sensor dies	4860	381
10% sensors die	5136	1704
30% sensors die	5225	3092

3.4 Conclusions

In this chapter, we have proposed a new flow-balanced routing (FBR) protocol for multi-hop clustered wireless sensor networks. In FBR, the cluster formation is performed only once at the beginning of the network operation and is determined on the basis of the the overlapping degrees of sensors. Some sensors whose sensing areas are covered by others are put into sleep mode in order to save energy. The cluster heads are constructed in a multi-level architecture with the sink at the top and there may be multiple paths from each head to the sink. On the basis of this novel network architecture, a flow-balanced routing algorithm is proposed to assign the flow from a head to the sink over multiple paths to equalize the power consumption of sensors. Furthermore, a local rerouting algorithm is proposed to reconfigure the network topology only at the location where any topological change occurs due to the dropouts of exhausted sensors.



(a) FBR



(b) CPCP

Figure 3.17: Examples of network topologies constructed in multi-hop approaches when sensors are regularly arranged in a lattice.

The proposed protocol, FBR, has been evaluated in comparison with previous protocols, LEACH, HEED, CPCP, and also two modified versions of HEED, i.e., M-HEED and HEED-FBR, using simulation. The results show that FBR yields longer network lifetime and also longer coverage lifetime than other protocols. The network lifetime of FBR can be more than five times longer than that of CPCP, the best among the previous protocols under consideration, and at the same time the coverage lifetime can be two times longer. Furthermore, the effects of the parameters have been examined with a wide range of parameter settings, and FBR always outperforms the others.

Chapter 4

Data Allocation in Sensor Clouds

4.1 System Model

The sensor data service model considered in this thesis is shown in Figure 4.1, which is composed of a number of cache nodes along with a few data sources each of which maintains all the data collected from a special sensor network. It is assumed that each sensor network generates only one type of data and the data is updated periodically. A user accesses the sensor data via his/her nearest cache node and can obtain the required data from the nearest cache node right away if it has the data. However, if the required data is not stored at the nearest cache node, it will be transferred, via the nearest cache node, from another nearby cache node that has the required data or from the data source. To better understand the data allocation model, let us see a scenario for climate data processing using the network model shown in Figure 4.1. All the climate data, e.g., the data collected by the National Oceanic and Atmospheric Administration (NOAA) [58], can be stored at a data source, say s_1, and need to be updated periodically. A cache node, say node i or j, can determine whether to cache the data depending on the user demand. A user, say user k, connected directly to node i can obtain his/her required data if the data is cached at node i or via node i from a nearby node, say node j, if node j has the data item. In the worst case, the data should be transferred from data source s_1 via node i. if there are no data copies in the cache network.

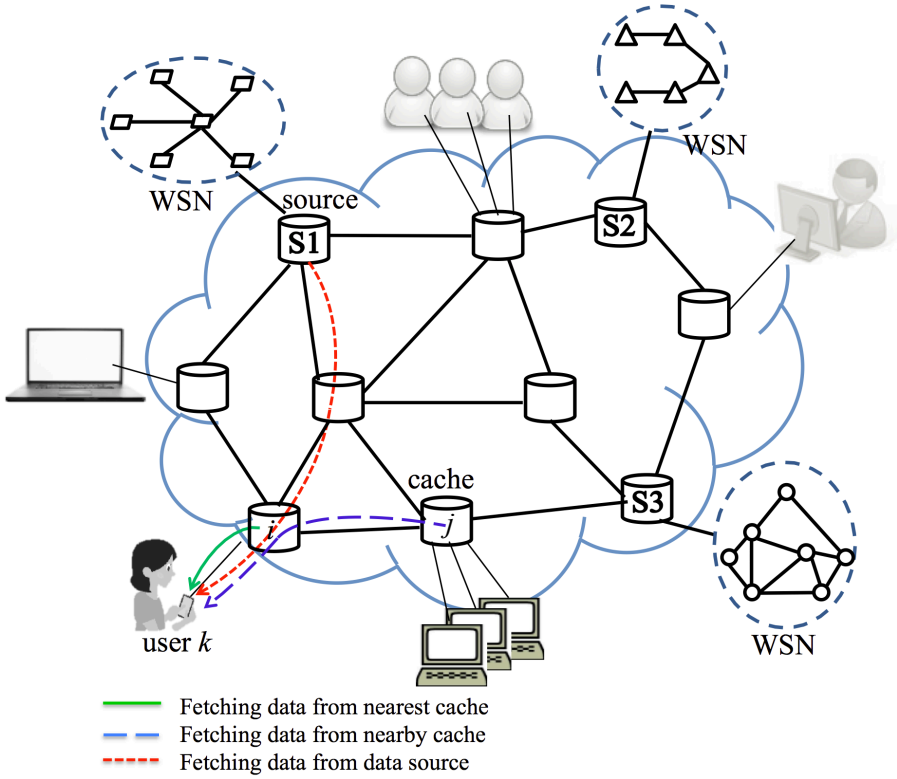


Figure 4.1: Sensor data service model.

A cache node can store only a limited amount of data and the determination for data cache locations depends on the user demand. For the sake of simplicity, it is assumed that the demand arriving at each cache node is given in advance. Three kinds of data operation costs are considered in this model: the *assignment cost* for transferring the data items periodically from their data sources to the cache nodes, the *placement cost* for storing the data items at the cache nodes, and the *access cost* for the users to obtain their required data. It is assumed that the shortest path between any two nodes in the network is known in advance. Our objective is to determine the best data allocation so as to minimize the total data operation costs.

The sets of cache nodes and the data sources are denoted by $\mathcal{N}$ and $\mathcal{S}$, respectively, and let $\mathcal{N}' = \mathcal{S} \bigcup \mathcal{N}$. The data access and maintenance model is illustrated in Figure 4.2, where data source s_k maintains data item k and node j may keep a copy of data item k. The users of node i, or simply denoted by node i, can obtain data item k from either node j or data source s_k. The demand of node i is denoted by h_i^k. The

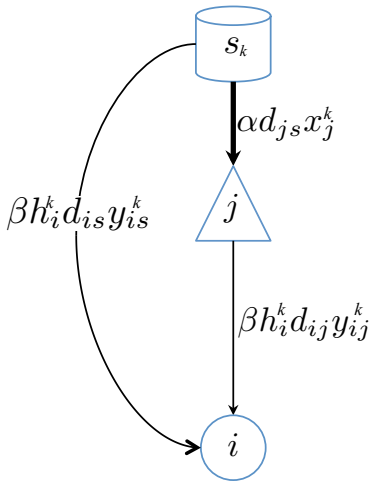


Figure 4.2: Data access and maintenance model.

cost for placing a data copy at node j is f_j, and the data transmission costs per unit data per unit distance from data source s_k to cache node j and from s_k or j to node i are denoted by α and β, respectively. Since the latency for data transmission from a data source to the cache nodes may not be so sensitive compared with that for a user request, it is assumed that $\alpha \leq \beta$. The distance between nodes i and j is denoted by d_{ij}. Two variables, $x_j^k (j \in \mathcal{N})$ and $y_{ij}^k (i \in \mathcal{N}, j \in \mathcal{N}')$, are used to indicate the cache allocation and the data access decisions, respectively. The value of x_j^k is set to 1 if node j has a copy of data item k and 0 otherwise. On the other hand, the value of y_{ij}^k is set to 1 if node i obtains data item k from node j and 0 otherwise. The symbols used in this chapter are listed in Table 4.1.

4.2 Single-type Data Allocation Problem (SDAP)

4.2.1 Formulation of SDAP

We first consider the single-type data allocation problem, where there is only one data item in the network, i.e., $|\mathcal{S}| = 1$. Here, the data source is denoted by s, and the demand of node i and the two decision variables are simply denoted by h_i, x_j, and y_{ij}, respectively. The data allocation problem with a single type data can be formulated

Table 4.1: Symbols used in this chapter.

$\mathcal{N}$	Set of cache nodes
$\mathcal{S}$	Set of data sources
$\mathcal{N}'$	Set of data sources and cache nodes, i.e., $\mathcal{N}' = \mathcal{S} \bigcup \mathcal{N}$
h_i^k	Demand of data k at node i
Q_j	Capacity of node j
f_j	Data placement cost at node j
d_{ij}	Distance between node i and j
α	Unit assignment cost
β	Unit access cost
x_j^k	Decision variable, $x_j^k = \begin{cases} 1, \text{if data } k \text{ is cached at node } j, \\ 0, \text{otherwise} \end{cases}$
y_{ij}^k	Decision variable, $y_{ij}^k = \begin{cases} 1, \text{if node } i \text{ accesses data } k \text{ from node } j, \\ 0, \text{otherwise} \end{cases}$

as the following integer programming problem:

$$(SDAP)$$

$$\min \quad \alpha \sum_{j \in \mathcal{N}} d_{sj} x_j + \sum_{j \in \mathcal{N}} f_j x_j + \beta \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}'} h_i d_{ij} y_{ij}, \tag{4.1}$$

subject to

$$\sum_{j \in \mathcal{N}'} y_{ij} = 1, \forall i \in \mathcal{N}, \tag{4.2}$$

$$x_j \geq y_{ij}, \forall i, j \in \mathcal{N}, \tag{4.3}$$

$$x_j \in \{0, 1\}, \forall j \in \mathcal{N}, \tag{4.4}$$

$$y_{ij} \in \{0, 1\}, \forall i \in \mathcal{N}, j \in \mathcal{N}'. \tag{4.5}$$

The objective function (4.1) minimizes the sum of the assignment cost, the placement cost and the access cost. Constraint (4.2) stipulates that node i gets a data from exactly one node, either the source node or a cache node. Constraint (4.3) means that the data requests of node i cannot be served by node j unless the data is cached at node j. Furthermore, constraints (4.4) and (4.5) are the integrality constraints on the decision variables. Note that if we do not consider the assignment cost, our problem can be reduced to an uncapacitated fixed charge facility location problem [55].

According to [57], we know that finding the optimal solutions for the above opti-

mization problem is NP-hard. Therefore, in this thesis we first solve this problem by relaxing constraint (4.2) and then choose only the nearest cache node for each data request.

4.2.2 A Lagrangian Relaxation Algorithm for SDAP

By relaxing constraint (4.2), i.e., by allowing a requesting node to be able to obtain a data item from more than one cache node, the problem (4.1) can be transformed to the following problem:

$$
\begin{aligned}
&(LR - SDAP) \\
&\max_{\lambda} \min_{x,y} \quad \sum_{j \in \mathcal{N}} (\alpha d_{sj} + f_j) x_j + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}'} (\beta h_i d_{ij} - \lambda_i) y_{ij} + \sum_i \lambda_i, \qquad (4.6) \\
&\text{subject to} \\
&\quad (4.3), (4.4), (4.5), \\
&\qquad \lambda_i > 0, \forall i \in \mathcal{N}, \qquad (4.7)
\end{aligned}
$$

where x, y, and λ denote the sets of $x_j (j \in \mathcal{N})$, $y_{ij} (i \in \mathcal{N}, j \in \mathcal{N}')$, and $\lambda_i (i \in \mathcal{N})$, respectively.

For a given set of the Lagrange multipliers, λ, we can find the optimal solutions of problem (4.6). At first, let us see the decision variables, y_{ij}. If $\beta h_i d_{ij} - \lambda_i \geq 0$, we can set $y_{ij} = 0$, and otherwise, we can set $y_{ij} = 1$. On the other hand, as shown in constraint (4.3), y_{ij} should not be greater than x_j. Thus, for each j, we can compute $V_j = \alpha d_{sj} + f_j + \sum_i \min\{0, \beta h_i d_{ij} - \lambda_i\}$, and if we set $x_j = 1$, the value of objective function (4.6) will be changed by the V_j. If $V_j < 0$, it is advantageous to set $x_j = 1$; otherwise, we should set $x_j = 0$. Since the optimal solutions of problem (4.6) for any fixed λ is not greater than the optimal solutions of primal problem (4.1) [57], we can use the value of objective function (4.6) as a *lower bound*, denoted by $LB(\lambda)$, of the primal problem. Furthermore, the optimal solution $x_j (j \in \mathcal{N})$ of problem (4.6) can be used as a feasible solution to the primal problem since it satisfies constraints (4.3) and (4.4). By assigning all the data requests of node i to the nearest cache node, we

can ensure constraint (4.2), i.e., $\sum_{j \in \mathcal{N}'} y_{ij} = 1$, holds. The obtained x and y is a feasible solution of the primal problem (4.1) and its corresponding value of object function (4.1) can be used as an *upper bound*, denoted by $UB(\lambda)$, of the primal problem.

In order to compute a good solution of the primal problem using the Lagrangian relaxation approach, we revise a sequence of Lagrange multipliers using a standard subgradient optimization procedure, as follows:

$$\lambda_i^{n+1} = \max\{0, \lambda_i^n - t^n(\sum_j y_{ij}^n - 1)\}, \tag{4.8}$$

where t^n is the nth iteration of the Lagrangian procedure, called stepsize, and

$$t^n = \frac{\gamma^n(UB(\lambda^n) - LB(\lambda^n))}{\sum_i(\sum_j y_{ij}^n - 1)^2}, \tag{4.9}$$

where y_{ij}^n and γ^n are the decision variables and a constant value in the nth iteration, respectively. Furthermore, $UB(\lambda^n)$ and $LB(\lambda^n)$ are respectively the upper and lower bounds of the objective function (4.1) in the nth iteration. If the value of $LB(\lambda^n)$ does not change after a certain number of iterations, say, a, we choose to decrease the value of λ, e.g., $\gamma^n = \omega\gamma^{n-a}, 0 < \omega < 1$. The initial values of λ_i and γ are given by $\lambda_i^1 = \min_{(i,j) \neq (i,i)} \beta h_i d_{ij}$ and $\gamma^1 = 2$, respectively.

Note that the Lagrangian relaxation method may not always lead to an optimal solution [59]; i.e., the difference between UB and LB may not vanish. Therefore, we should stop the iterations when the difference between UB and LB becomes less than a tolerance error ϵ_0 or after a certain number of iterations. The Lagrangian relaxation approach (indicated by LAG) is detailed in Algorithm 5 and its computation complexity is given by $\mathcal{O}(I|\mathcal{N}|^2)$, where I denotes the maximum number of iterations.

4.2.3 A Greedy Algorithm for SDAP

In the proposed greedy algorithm (indicated by GRE), whether to place a data copy at a node or not is determined by the costs of data placement and data transmission from the data source or another data cached node to the node. The nodes next to

Algorithm 5 Lagrangian relaxation algorithm for SDAP

1: **Initialization**: $n = 0, UB = UB(\lambda^n) = \infty, LB = LB(\lambda^n) = 0$
2: **while** $UB - LB \geq \epsilon_0$ and $n \leq I$ **do**
3: $n \leftarrow n + 1$
/* step 4 $\sim$ 13: calculate the LB by solving the relaxed problem (4.6) */
4: **for** each node $i \in \mathcal{N}$ **do**
5: Set Lagrange multipliers λ_i^n according to (4.8)
6: **for** each candidate cache node $j \in \mathcal{N}$ **do**
7: Calculate V_j $(V_j = \alpha d_{sj} + f_j + \sum_i \min\{0, \beta h_i d_{ij} - \lambda_i^n\})$
8: Set $x_j^n = \begin{cases} 1, \text{if } V_j < 0, \\ 0, \text{otherwise} \end{cases}$
9: Set $y_{ij}^n = \begin{cases} 1, \text{if } x_j^n = 1 \text{ and } \beta h_i d_{ij} - \lambda_i^n < 0, \\ 0, \text{otherwise} \end{cases}$
10: **end for**
11: **end for**
12: Calculate the value of the objective function (4.6) using x^n, y^n, λ^n, and let it to be $LB(\lambda^n)$
13: **if** $LB(\lambda^n) > LB$ **then** $LB \leftarrow LB(\lambda^n)$
/* step 14 $\sim$ 19: calculate the feasible solution of problem (4.1) and the UB */
14: **for** each node $i \in \mathcal{N}$ **do**
15: Set $x_j^{'n} = x_j^n$
16: Set $y_{ij}^{'n} = \begin{cases} 1, \text{if } x_j^{'n} = 1 \text{ and } j \text{ is the nearest cache node of } i, \\ 0, \text{otherwise} \end{cases}$
17: **end for**
18: Calculate the value of objective function (4.1) using $x^{'n}$, $y^{'n}$, and let it to be $UB(\lambda^n)$
19: **if** $UB(\lambda^n) < UB$ **then** $UB \leftarrow UB(\lambda^n)$, $x \leftarrow x^{'n}$, $y \leftarrow y^{'n}$
20: **end while**
21: Let x, y to be the solution of primal problem (4.1)

the data source trigger the algorithm and then inform their caching decisions to the neighboring nodes. The data allocation is determined according to a breadth-first order, i.e., a node closer to the data source has a higher priority to make the cache decision. After a node decides whether to cache the data, it is marked as a *determined node*. The set of the determined nodes is denoted by N_c. The distance between an undetermined node, say node i, and the nodes in N_c, denoted by d_i^*, is defined by the shortest distance from node i to the closest node in N_c, i.e., $d_i^* = \min_{j \in N_c} d_{ij}$.

Initially, only the source node s is included in N_c. All the undetermined nodes in $\mathcal{N}$ are numbered in increasing order using a breadth-first search (BFS) algorithm. For

the nodes with the same hop to the data source, a node with a larger h_i has a higher priority to make the cache decision. We know that the cost for caching a data copy at node i is $f_i + \alpha d_{si}$. If node i decides to cache a data copy, the nodes in the range of $\frac{1}{2}d_i^*$ from node i, which is denoted by N_i, will not place a data copy and access the data copy from node i if necessary. We treat the sum of the costs for accessing the data from node $m(m \in N_i)$ as the benefit for data placement at node i, and node i will cache the data if only the benefit is larger than the placement cost, i.e.,

$$\sum_{m \in N_i} \frac{1}{2}\beta h_m d_i^* > f_i + \alpha d_{si} \tag{4.10}$$

Note that node i is also included in N_i.

Algorithm 6 Greedy algorithm for SDAP

```
Initialization: N_c ← s, sort 𝒩 using BFS, i = 1
while i ≤ |𝒩| do
    d_i^* = min_{j∈N_c} d_ij
    if Σ_{m∈N_i} (1/2)βh_m d_i^* > f_i + αd_si then
        Set x_i = 1
        Add i to N_c
    else
        Set x_i = 0
    end if
    i ← i + 1
end while
```

From relation (4.10), we see that node i makes the cache decision considering the surrounding nodes within a distance, which is bound by $\frac{1}{2}d_i^*$. In a general case, we have $|N_i| \ll |\mathcal{N}|$. The proposed greedy algorithm is detailed in Algorithm 6, and its computation complexity is given by $\mathcal{O}(\max\{|N_i|\}|\mathcal{N}|)$.

4.3 Multi-type Data Allocation Problem (MDAP)

It is common to have more than one different data items and in this thesis we consider two versions of the multi-type data allocation problem: uncapacitated and capacitated.

4.3.1 Uncapacitated Multi-type Data Allocation Problem (UMDAP)

Generally, a cache node has a limited storage capacity, if the capacity of a cache node is greater than the amount of sensor data, we can consider the data allocation problem as an uncapacitated multi-type data allocation problem; that is, there is no capacity limitation. Therefore, the data items can be treated independently with one another and the data allocation problem can be formulated as follows.

$$
\begin{aligned}
&(UMDAP) \\
&\min \quad \sum_{k\in\mathcal{S}}(\alpha\sum_{j\in\mathcal{N}} d_{sj}x_j^k + \sum_{j\in\mathcal{N}} f_j x_j^k + \beta\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}'} d_{ij}h_i^k y_{ij}^k) && (4.11)\\
&= \min \quad \alpha\sum_{j\in\mathcal{N}} d_{sj}x_j^1 + \sum_{j\in\mathcal{N}} f_j x_j^1 + \beta\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}'} d_{ij}h_i^1 y_{ij}^1 \\
&+ \min \quad \alpha\sum_{j\in\mathcal{N}} d_{sj}x_j^2 + \sum_{j\in\mathcal{N}} f_j x_j^2 + \beta\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}'} d_{ij}h_i^2 y_{ij}^2 \\
&\qquad \vdots \\
&+ \min \quad \alpha\sum_{j\in\mathcal{N}} d_{sj}x_j^{|\mathcal{S}|} + \sum_{j\in\mathcal{N}} f_j x_j^{|\mathcal{S}|} + \beta\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}'} d_{ij}h_i^{|\mathcal{S}|} y_{ij}^{|\mathcal{S}|}, && (4.12)\\
&\text{subject to} \\
&\sum_{j\in\mathcal{N}'} y_{ij}^k = 1, \forall i\in\mathcal{N}, k\in\mathcal{S}, && (4.13)\\
&x_j^k \geq y_{ij}^k, \forall i,j\in\mathcal{N}, k\in\mathcal{S}, && (4.14)\\
&x_j^k \in \{0,1\}, \forall j\in\mathcal{N}, k\in\mathcal{S}, && (4.15)\\
&y_{ij}^k \in \{0,1\}, \forall i\in\mathcal{N}, j\in\mathcal{N}', k\in\mathcal{S}. && (4.16)
\end{aligned}
$$

This problem can be solved for each data item separately by using the extended Lagrangian relaxation algorithm or the greedy caching algorithm described in section 4.2.

4.3.2 Capacitated Multi-type Data Allocation Problem (CMDAP)

When the capacity of each cache node is less than the amount of sensor data, the total number of data items stored at node j is bound by its capacity Q_j. Thus, the data allocation problem is formulated as follows.

$$
\begin{aligned}
&(CMDAP)\\
&\min \alpha \sum_{k\in\mathcal{S}}\sum_{j\in\mathcal{N}} d_{sj}x_j^k + \sum_{k\in\mathcal{S}}\sum_{j\in\mathcal{N}} f_j x_j^k + \beta \sum_{k\in\mathcal{S}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}'} d_{ij}h_i^k y_{ij}^k, && (4.17)\\
&\text{subject to}\\
&\quad (4.13), (4.14), (4.15), (4.16),\\
&\quad \sum_{k\in\mathcal{S}} x_j^k \le Q_j, \forall j \in \mathcal{N}. && (4.20)
\end{aligned}
$$

The constraint (4.20) shows that the number of data items cached at node j can not exceed its capacity. By relaxing capacity constraint (4.20) using Lagrangian multipliers μ_j, we have

$$
\begin{aligned}
&\max_{\mu}\min_{x,y} \sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{S}}(\alpha d_{sj} + f_j - \mu_j)x_j^k + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}'}\sum_{k\in\mathcal{S}} \beta h_i^k d_{ij} y_{ij}^k + \sum_{j\in\mathcal{N}} \mu_j Q_j, && (4.21)\\
&\text{subject to} \qquad \mu_j \ge 0, \forall j \in \mathcal{N}'.
\end{aligned}
$$

By substituting $\hat{f}_j = (\alpha d_{sj} + f_j - \mu_j)$ and $\hat{Q} = \sum_j \mu_j Q_j$ into relation (4.21), we obtain the following objective function for the fixed values of μ_j:

$$
\begin{aligned}
&\max_{\mu}\min_{x,y} \sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{S}} \hat{f}_j x_j^k + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}'}\sum_{k\in\mathcal{S}} \beta h_i^k d_{ij} y_{ij}^k + \hat{Q}_j, && (4.22)\\
&\text{subject to}\\
&\quad (4.13), (4.14), (4.15), (4.16).
\end{aligned}
$$

We see that the problem (4.22) can be solved similarly to the UMDAP problem if the values of $\mu_j (j \in \mathcal{N})$ are fixed. However, it is generally difficult to determine the value of μ_j since the solutions of different data allocations affect one another.

Therefore, it is difficult to apply the Lagrangian relaxation approach to this problem. Instead, we consider the UMDAP problem as a special case of CMDAP wherein the maximum number of data items is less than the capacity of each node. The lower bound of UMDAP can also be used as the lower bound of CMDAP, even though it may be over conservative and difficult to realize.

In this thesis, we propose a heuristic algorithm, shown in Algorithm 7, to solve the CMDAP. We first run either the Lagrangian relaxation algorithm (Algorithm 5) or the greedy caching algorithm (Algorithm 6) for each data item $k \in \mathcal{S}$ without considering the capacity constraints. For each node $j (j \in N_v)$ whose capacity constraint is violated, we first check the cost for reallocating each data item and then determine to reallocate the data with the minimum cost accordingly. Here, we examine each of the neighboring nodes of node j, denoted by the set N_j^d, within a predefined distance d to determine how much the total data allocation cost increases. If we find the best node in N_j^d, denoted by j_*^k, for data k, the reallocation is performed and the total allocation cost is updated. This procedure is repeated until the capacity constraint violation is alleviated or no candidate can be found. If we cannot find a neighboring node to cache data k, there will be no copy of data k in node j or in any neighbor of j in N_j^d. The time complexity of the heuristic algorithm is given by $\mathcal{O}(|\mathcal{S}||\mathcal{N}|)$.

4.4 Performance Evaluation

In this section, we evaluate the performance of our proposed algorithms using numerical experiments. Following the typical service charges provided in Amazon S3 [60], we set the placement cost $f_i = 1 (i \in \mathcal{N})$, the assignment cost $\alpha = 3$, and the access cost $\beta = 4$. It is assumed that the data is transmitted along the shortest path between two nodes. The parameter settings used in the experiments are shown in Table 4.2. In order to evaluate our proposed algorithms, we define a performance metric called the *performance gap* (PG) between the objective function value (OV), obtained by using our proposed algorithms and the lower bound (LB) of the objective functions

Algorithm 7 Reallocation Heuristic Algorithm for CMDAP

```
run a data allocation algorithm, e.g., the greedy algorithm proposed in this thesis,
to allocate data items without considering capacity constraints
let the set of nodes violating capacity constraints to be N_v and the set of data
items allocated to node j (j ∈ N_v) to be S_j
for each node j ∈ N_v do
  while |S_j| > Q_j do
    find node j's neighbors, N_j^d, that are located within d hops from j and |S_i| ≤
    Q_i(i ∈ N_j^d)
    if N_j^d = ∅ then
      delete item k from node j, i.e., S_j ← S_j \ k, such that the value increment
      of function (19) is the minimum
    else
      find a node, denoted by j_*^k, to reallocate item k such that the value incre-
      ment of function (19) is the minimum
      allocate item k to j_*^k, i.e., S_{j_*^k} ← S_{j_*^k} + 1, and delete k from node j, i.e.,
      S_j ← S_j \ k
    end if
  end while
end for
```

as follows:

$$PG = \frac{OV - LB}{LB}. \tag{4.23}$$

Note that generally the lower bound cannot be realized and the PG indicates only how close to the theoretical lower bound an allocation solution could be.

Table 4.2: Primary parameters used in experiments.

Parameter	Value	Description
f_j	1	Fixed placement cost, $j \in \mathcal{N}$
α	3	Unit assignment cost
β	4	Unit access cost
h_i^k	1	Demand, $i \in \mathcal{N}$, $k \in \mathcal{M}$
ϵ_0	1.0×10^{-5}	Tolerance error (in Lagrangian relaxation algorithm)
I	30,000	Max number of iterations (in Lagrangian relaxation algorithm)

4.4.1 Numerical Experiments for SDAP

We first examined the extended Lagrangian relaxation algorithm (LAG) and the greedy caching algorithm (GRE) for SDAP. A well-known integer programming solver called CPLEX was used to compare with our proposed algorithms. An arbitrary network model shown in Figure 4.3 is first used in our experiments. In this network model, there is one data source, denoted by node 0, and 9 cache nodes. Data allocation results obtained by CPLEX, LAG, and GRE are shown in Figure 4.3, and the corresponding costs are 45, 45, and 48, respectively. The LB value obtained by the Lagrangian algorithm is 44.76. In this experiment, both the CPLEX and LAG yield the optimal solutions, and the GRE obtained a feasible solution with the PG value of 7.24%. The execution times of these three approaches are 20ms, 10ms, and 2ms, respectively.

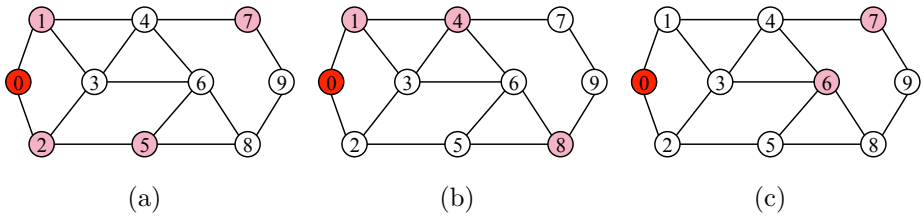


Figure 4.3: Data allocation results for SADP: (a) CPLEX solver, (b) Lagrangian relaxation algorithm, (c) Greedy algorithm.

To further investigate the performance in a larger network, we performed experiments using a lattice network model with 16×16 nodes as shown in Figure 4.4. Each node has at most four neighboring nodes and the data source is located at node 0. Note that our proposed approaches are applicable to arbitrary topology, and we used the lattice network model in the experiments simply for easy understanding. We compared the computation times of LAG and GRE with CPLEX and a random caching approach (indicated by RAN). In order to improve the convergence speed of CPLEX we relaxed the constraint $y_{ij} \in \{0, 1\}$ by a weaker constraint $0 \leq y_{ij} \leq 1$ to get a feasible solution x_j and then assigned each demand node to access data from the nearest cache node. Therefore, the obtained solution is optimal. The solution obtained by this method and the original problem are respectively indicated by CPLEX_LP and

CPLEX_IP in the figures. In the random approach, a given percentage of nodes are randomly selected to place data copies and the results shown in the figures are obtained by placing the data copies to 5%, 10%, and 20% of nodes (indicated by RAN5, RAN10, and RAN20 in the figures), respectively.

The experiment results, including the objective values, computation time, and performance gaps, are shown in Table 4.3. In this experiment, the lower bound value calculated by the LAG is $2,475$. The convergence speeds of the CPLEX_IP and CPLEX_LP are much slower than others. The CPLEX_IP could not yield the optimal solution after 10-hour computation in this experiment, and therefore we stopped the computation and selected the best solution with a value of 2,506 obtained in the 10-hour computation period for comparison. The CPLEX_LP, by contrast, obtained the optimal solution with a value of 2,494 but spent nearly one hour to converge. The random approach with 10% data caching, RAN10, yields total cost of $3,043$ but with only a computation time of 10ms. From our experiments, we can see that LAG and GRE have distinct advantages in convergence speed compared with the CPLEX and in optimality compared with the random approach. Furthermore, the performance difference between LAG and the optimal solution obtained by CPLEX is less than 3%.

Table 4.3: Results of various approaches for SDAP in 16×16 lattice network model.

	Cost	Time	PG
LAG	2,565	22s	3.63%
GRE	2,842	20ms	14.83%
CPLEX_IP	2,506	>10h	1.25%
CPLEX_LP	2,494	0.9h	0.77%
RAN10	3,043	10ms	22.95%

The data allocation results obtained by the LAG, GRE, and CPLEX_LP are displayed in Figure 4.4. Contrary to our intuition that data would be placed evenly in the network, we obtained the results that nodes near the sink cached more data than that faraway from the sink, some adjacent nodes near the sink even cached data at the same time. For example, neighboring nodes 1, 2, and 3 cached data at the same time in Figure 4.4(a). That is because the data service model considers the data as-

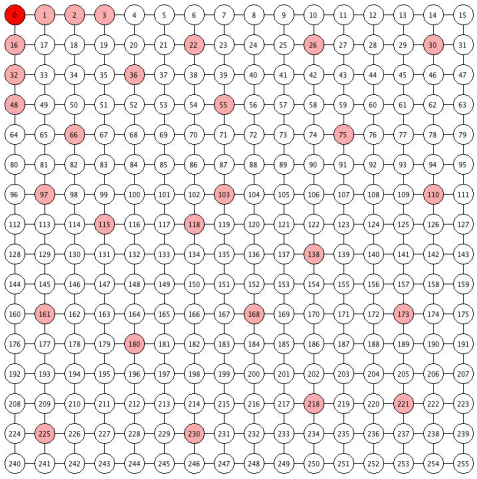
signment cost generated by periodically updating data from the source node to each cache node. The further a cache node is from the source, the higher assignment cost it needs. So there were more data copies near the source node. The following Figure 4.4(c) illustrating an optimal solution also has this characteristic.

We also examined the effect of the user demand on the performance of the data allocation. Figure 4.5 illustrates the costs obtained by using the algorithms under consideration. From this figure, we see that the costs obtained by using RAN10, where 10% of nodes are randomly selected to cache data items, increases linearly corresponding to the user demand, while costs obtained using the Lagrangian relaxation algorithm, the greedy algorithm and CPLEX_LP converge to the lower bound. It is interesting to see that both the Lagrangian relaxation algorithm and the greedy algorithm behave similarly.

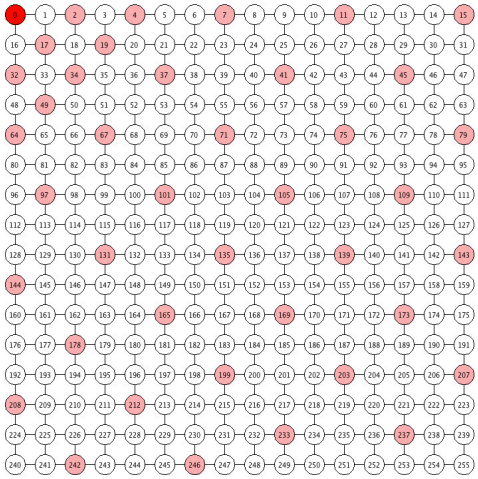
4.4.2 Numerical Experiments for MDAP

In the multi-type data allocation problem, we consider the same 16×16 lattice network model as shown in Figure 4.4 with 4 data items. We assumed that each node can only cache one data item; i.e., $Q_j = 1$ $(j \in \mathcal{N})$. The four data sources are placed at the corner nodes of the lattice network, and are indicated by the red, blue, green, and yellow colors, respectively. The copies of the four data items are indicated by the light red, blue, green, and yellow colors, respectively. For UMDAP, we see that the total data allocation cost is the sum of the costs for allocating all the four data items independently. Therefore, the lower bound (LB) of UMDAP can also be considered the sum of the lower bound for allocating each data item. As shown in Figure 4.9, the value of LB is $9,900$. As mentioned earlier in section 4.4, the LB is generally difficult to realize and therefore it is used only for comparison purposes. We did not use CPLEX_IP to calculate solutions for multi-type data allocation model, since its computation time is unrealistically long.

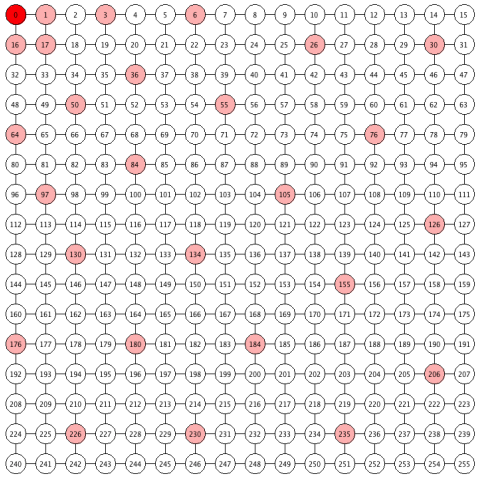
We first calculated the solutions for UMDAP, that is, we determined the data allocation by ignoring the node capacity limitations. The data allocation results for UMDAP are shown in Figure 4.6. In the figure, a node with more than one



(a)



(b)



(c)

Figure 4.4: Data allocation results for SDAP in a 16×16 lattice network: (a) Lagrangian relaxation algorithm, (b) Greedy algorithm, (c) CPLEX_LP.

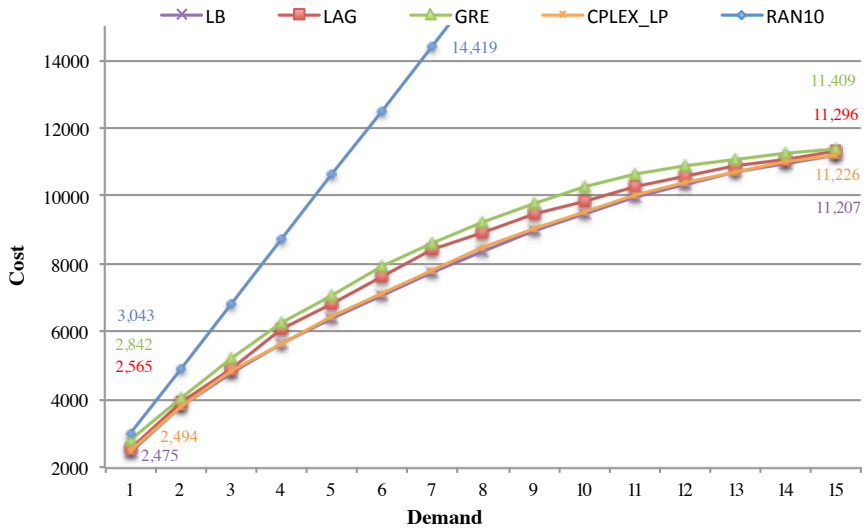
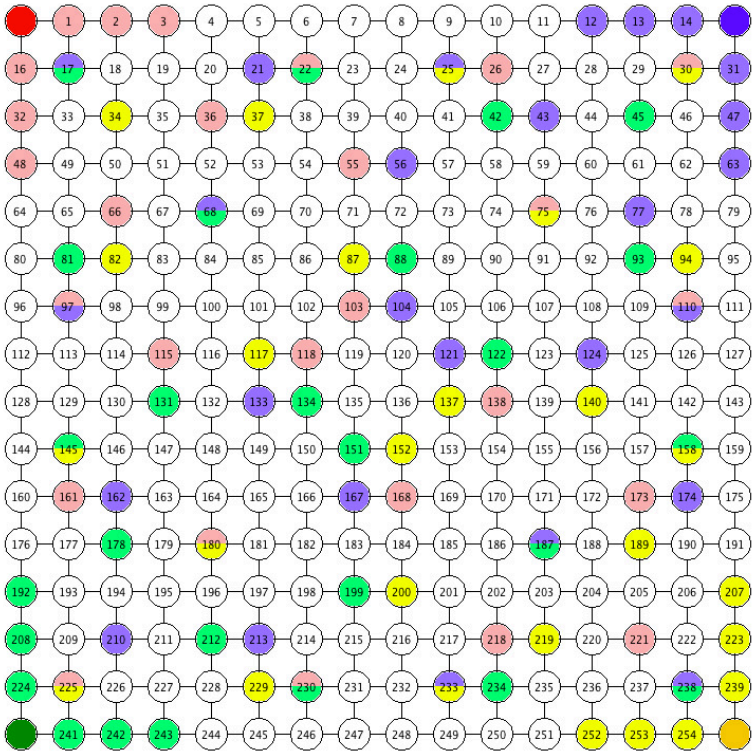


Figure 4.5: Performance comparison for various demands.

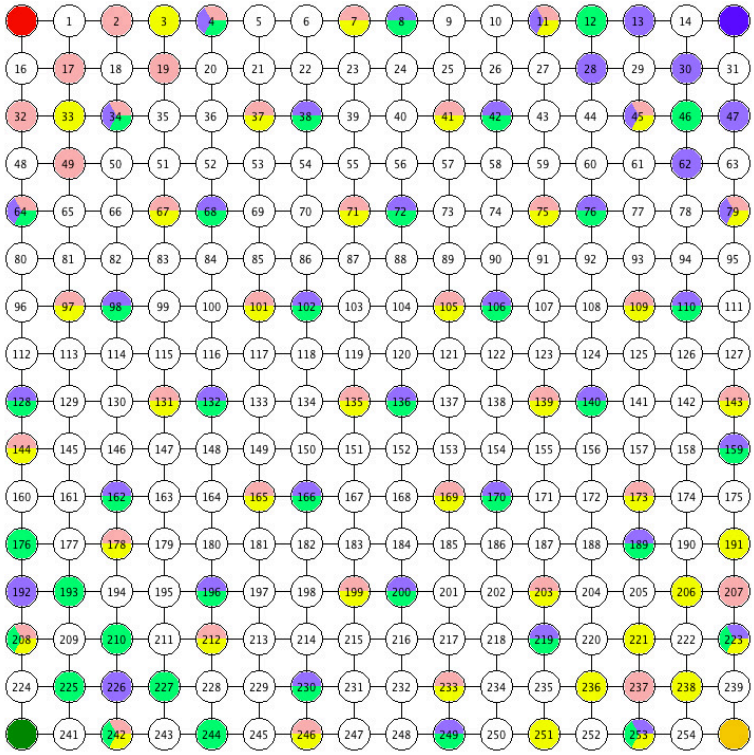
color means it holds more than one data copy. The solutions of UMDAP obtained by the Lagrangian relaxation algorithm and the greedy algorithm are indicated by UMDAP_LAG and UMDAP_GRE as shown in Figure 4.6 and they yielded the cost values of $10,260$ and $11,368$. Note that the performance gaps (PGs) of those two solutions are 3.64% and 14.83%, respectively.

Then, we executed our proposed heuristic algorithm (Algorithm 7) based on the solutions obtained for UMDAP to reallocate the data items at the nodes which capacity constraints are violated. The solutions obtained using the heuristic algorithm (Algorithm 7) are indicated by CMDAP_LAG and CMDAP_GRE, respectively, as shown in Figure 4.7. The solution values obtained by using the heuristic algorithm are $10,292$ and $11,228$, and the performance gaps (PGs) of the two solutions are 3.96% and 13.41%, respectively. Note that the solutions obtained by the heuristic algorithm are comparable to those when the node capacity limitation is ignored.

Since the CPLEX_LP could not yield an optimal solution after 24-hour computation, we stopped the computation and used the best solution with a value of 10,002 obtained in the 24-hour computation period for comparison. The data allocation results obtained by using the random approach, say, RAN5, RAN10, and RAN20, are shown in Figure 4.8 and the corresponding costs are $13,692$, $12,792$, and $14,599$, re-

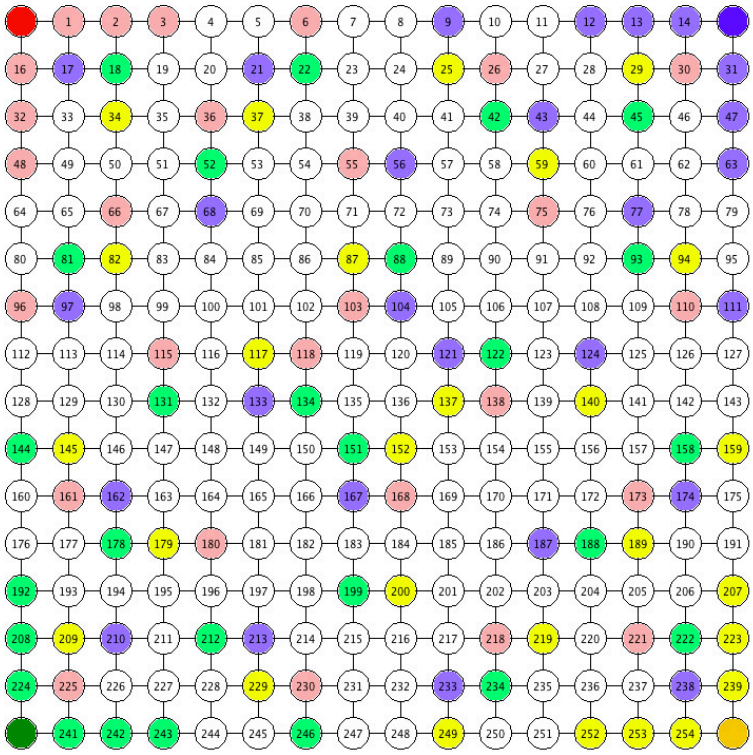


(a)

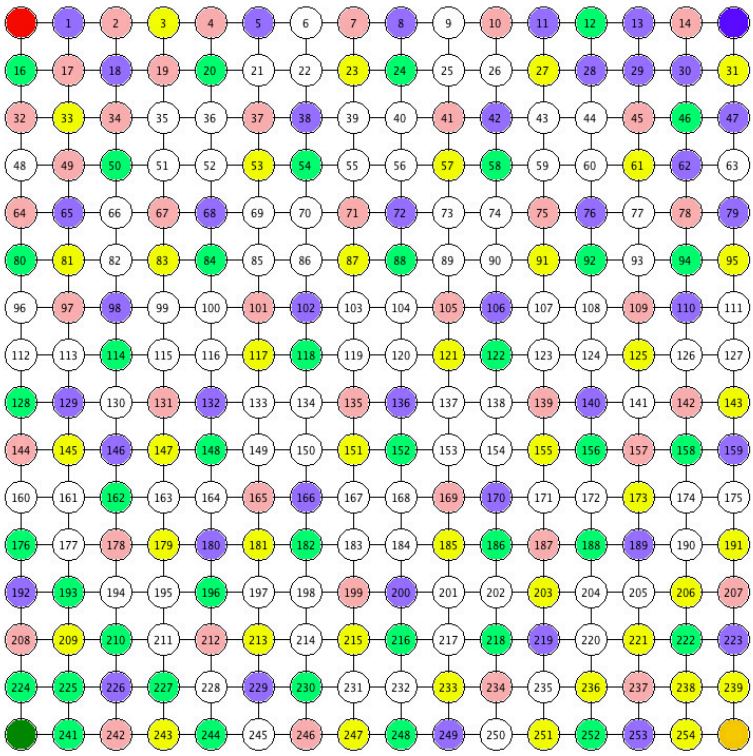


(b)

Figure 4.6: Data allocation results for UMDAP in a 16×16 lattice network: (a) Lagrangian relaxation algorithm, (b) Greedy algorithm.

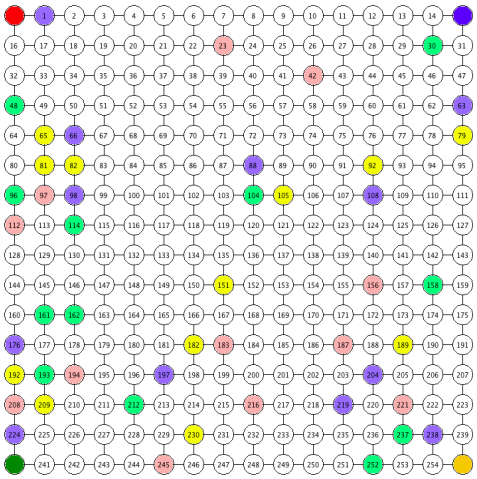


(a)

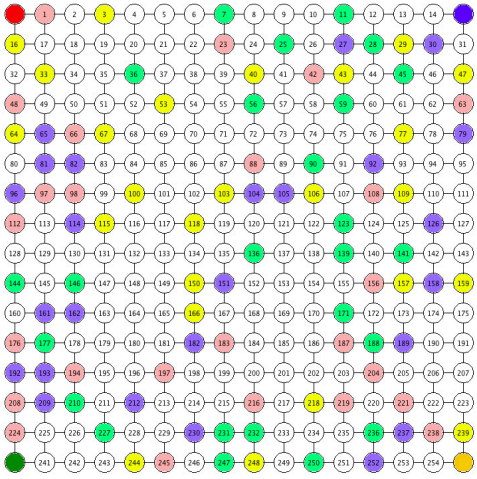


(b)

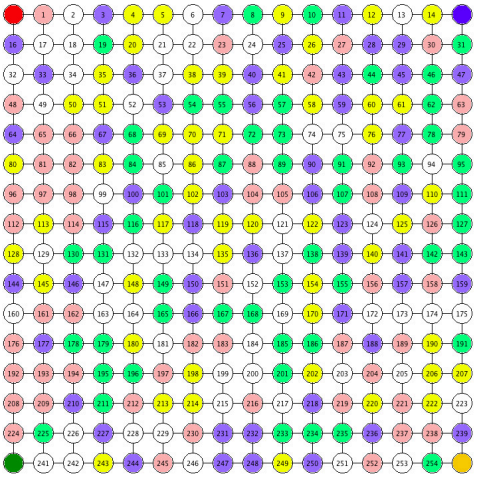
Figure 4.7: Data allocation results for CMDAP in a 16×16 lattice network: (a) Lagrangian relaxation algorithm, (b) Greedy algorithm.



(a)



(b)



(c)

Figure 4.8: Data allocation results for a random allocation approach in a 16×16 lattice network: (a) 5% nodes cache data, (b) 10% nodes cache data, (c) 20% nodes cache data.

Table 4.4: Results of various approaches for MDAP in 16×16 lattice network model.

	Cost	Time	PG
UMDAP_LAG	10,260	105.43s	3.63%
CMDAP_LAG	10,292	105.66s	3.96%
UMDAP_GRE	11,368	1.13s	14.83%
CMDAP_GRE	11,228	1.60s	13.41%
CPLEX_LP	10,002	$> 24h$	1.03%
RAN5	13,692	20ms	38.30%
RAN10	12,792	20ms	29.21%
RAN20	14,599	20ms	47.46%

spectively, as shown in Table 4.4, and the performance gaps (PGs) of these solutions are 38.30%, 29.21%, and 47.46%, respectively. We see that only 10% of nodes holding data copies yields the best performance. The costs of MDAP obtained by approaches used in our experiments are shown in Figure 4.9. We also see that the performance difference between CMDAP_LAG and LB is less than 4% and that CMDAP_LAG performs much better than RAN10 by 25%. Furthermore, we see that even though the computation time of CMDAP_LAG is much longer than the random approach but it still falls within the range of reality.

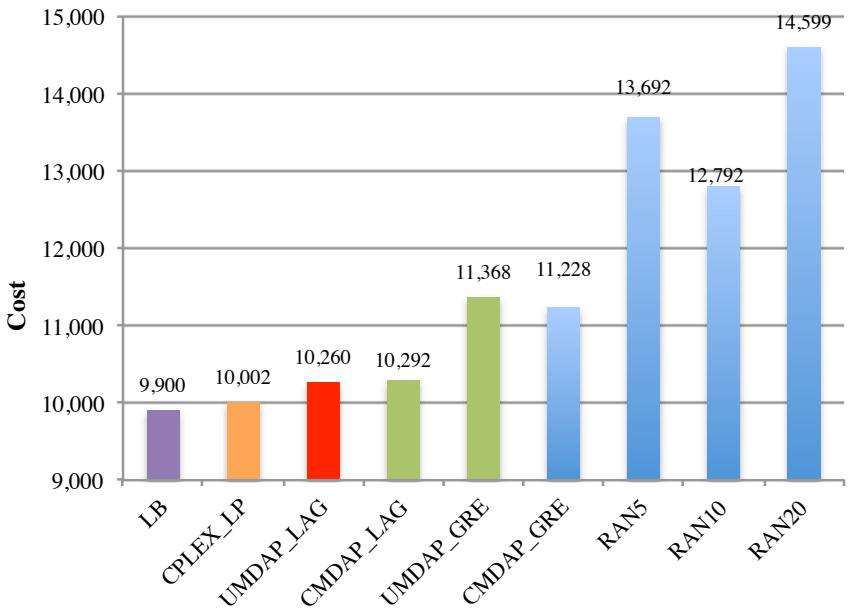


Figure 4.9: Data allocation costs with multiple data types.

4.5 Conclusions

In this chapter we proposed to place a number of cache nodes in the edge networks to store the frequently accessed data for sensor-clouds. We formulated a data allocation problem with a single data type, SDAP, and two data allocation problems with multiple data types, UMDAP, and CMDAP. In those problems, the data assignment cost, data placement cost, and the data access cost are taken into account. We first extended the traditional Lagrangian relaxation algorithm and then proposed a greedy caching algorithm to solve the SDAP problem. These two algorithms were further extended to solve the UMDAP and CMDAP. We compared our proposed algorithms with a well-known integer programming solver, CPLEX, and a simple random data allocation approach. The experiment results show that in the case with a single data type the performance of our extended Langrangian relaxation algorithm for SDAP is only less than 3% worse than CPLEX but the computation time is extremely shorter. Furthermore, in the case with multiple data types the performance of our proposed heuristic algorithm for CMDAP is less than 5% worse than the lower bound (LB) and is similar to the case where we ignore node capacity limitation. Furthermore, the computation time of the algorithm for CMDAP falls in a practical range, say, less than several seconds.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis has explored both the data aggregation in WSNs and the data allocation in sensor-clouds. The subject that the former faces is to design an energy-efficient data transmission protocol to prolong the lifetime of a WSN as long as possible, and that the latter faces is to find an efficient data allocation approach to control the data access and maintenance costs as low as possible.

In this thesis we have proposed a novel flow-balanced routing (FBR) protocol, that yields longer network lifetime and also longer coverage lifetime than previous works. Following the FBR, sensors are grouped into clusters according to the overlapping degrees, and then a multi-level backbone, not necessarily a tree, with the sink at the top level and the cluster heads at lower levels is constructed. The sensed data are transferred from the sensors to the sink through this multi-level backbone, and the energy imbalance between sensors can be alleviated by assigning the data flow over multiple paths. Furthermore, the overhead of network construction and maintenance can be greatly reduced by reconfiguring network locally. FBR considers both the power efficiency and coverage preservation at the same time. According to our experiment results, the FBR, compared with the previous work CPCP, which is the best among the previous protocols under consideration, yields more than five times longer network lifetime and more than two times longer coverage lifetime. Furthermore, the

effects of the parameters have been examined with a wide range of parameter settings, and FBR always outperforms the others.

The data allocation problem proposed in this thesis is the first work that focuses on improving the sensor data service experiences for users in sensor-clouds. The data service model proposed in this thesis takes account into factors such as data operation costs, overall network topology which consists of a set to service nodes in the edge network of a cloud, the demands of users for sensor data, the storage capacity of each service node. The data operation costs include the cost to assign sensor data collected from various WSNs to data service nodes in the sensor-cloud environment, the cost to maintain the data cached at the service nodes, and the cost to convey data from service nodes to the end users. A data allocation problem with a single data type, SDAP, and two data allocation problems with multiple data types, UMDAP, and CMDAP are formulated. And then an algorithm extended the traditional Lagrangian relaxation method and a greedy caching algorithm are proposed to solve the SDAP. These two algorithms were further extended to solve the UMDAP, and another heuristic algorithm is proposed to solve the CMDAP. Our proposed algorithms are compared with a well-known integer programming solver, CPLEX, and a simple random data allocation approach. The experiment results show that in the case with a single data type the performance of our extended Langrangian relaxation algorithm for SDAP is only less than 3% worse than CPLEX and the computation time is extremely shorter. Meanwhile, in the case with multiple data types the performance of our proposed heuristic algorithm for CMDAP is less than 5% worse than the lower bound (LB) and is similar to the case where we ignore node capacity limitations. More importantly, the computation time of the algorithm for CMDAP falls in a practical range, say, less than several seconds.

In conclusion, this thesis covers the sensor data aggregation in a WSN and data allocation in a sensor-cloud environment. We provide solutions to collect sensor data from various sensor networks energy efficiently and then allocate the collected data in the edge network of a cloud to provide data as a service for various users. Both subjects of this thesis possess an evident universality, which can be applied to a wide

variety of applications.

5.2 Future Work

Though the data aggregation in WSNs with fixed sinks and data allocation in cloud environments have been discussed in this thesis, some potential research topics such as follows are worthy of study in the future.

- Data aggregation in WSNs with mobile sinks.

  Recent years, WSNs with mobile sinks have attracted a lot of research activities [61, 62, 63, 64]. It is assumed that a mobile sink can be carried by a public surface transportation vehicle (e.g., a car, a bus) that repeatedly passes fixed trajectories in sensor fields to collect data. In this way, each sensor node can send data to the mobile sink over a short distance so that the energy consumption can be controlled. However, the considerable delay in collecting sensed data becomes a major disadvantage of using a mobile sink, since a node needs to wait for the sink to approach it. Finding an efficient moving scheme for a mobile sink also becomes one of the most important issues for this situation. At the same time, efficient collaboration methods between sensors, power management of sensors, etc., also need to be considered accordingly.

- Cooperative data processing among cache servers and end users.

  In the data service model proposed in this work, data can be provided as services to the end users after the data allocation phase. However, extended research such as cooperative data processing among cache servers and end users also can be conducted to improve the sensor data services experience for users. An example is the computation offloading between servers and end users, which mainly aims at alleviating limitations of battery capacity and computing capability of end users, especially the mobile users. Both component-partition and component-migration are two major issues in the computing offloading. The former aims at find the optimal component partition for a certain computation

task, while the latter aims at deciding which components should be offloaded to the servers such that the task can be completed at the minimal execution cost. Some related works such as [65, 66, 67, 68, 69, 70] have been performed in recent years. One of our researches focusing on the component-migration problem also has been conducted and the preliminary results can be found in [71].

- Real-world application system construction and management.

  With the development of wireless sensing and cloud computing technologies, the applications of WSNs are going to become more ubiquitous. In the real applications, the basic research findings we have obtained in this study can be used, and the specific problems in specific applications should be further considered. Combing the current situation of information and communications technology (ICT), the applied research in the future will be carried out on basic social infrastructure such as: 1) Application in medical treatment which increases chances of providing better medical health care services to patients and reducing costs for the medical institutions. In this area, high degrees of reliability and security are particularly important since there may be risks of life or privacy leaks [72, 73, 74]. 2) Application in traffic management. Sensors are set into vehicles or on roads to obtain the running statuses of vehicles. Features such as mobility, heterogeneity, and pluralism present challenges for traffic management system [75, 76]. 3) Application in electric power systems. WSNs are considered to provide cost-effective sensing and communication solution in power systems, including power generation, power delivery, and power utilization. However, harsh and complex electric-power-system environments pose great challenges in the reliability [77, 78].

Bibliography

[1] Agricultural sensors: improving crop farming to help us feed the world, http://www.dw.com/en/agricultural-sensors-improving-crop-farming-to-help-us-feed-the-world/a-17733350.

[2] M.V. Ramesh, "Design, development, and deployment of a wireless sensor network for detection of landslides," *Ad Hoc Networks* Vo1. 13, Part A, pp. 2-18, February 2014.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks a Survey," *Computer Networks*, Vo1. 38, No. 4, pp. 393-422, March 2002.

[4] Sensor-Cloud, http://sensorcloud.com/system-overview.

[5] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68-73, 2009.

[6] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, "A Survey on Sensor-Cloud: Architecture, Applications, and Approaches," *International Journal of Distributed Sensor Networks*, Vol. 2013, ID 917923, 18 pages, 2013.

[7] Sensor, http://whatis.techtarget.com/definition/sensor.

[8] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," *Proc. ACM Mobi- Comf99*, pp. 263-270, 1999.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proc. ACM MobiComf00*, pp. 56-57, 2000.

[10] I. Dietrich and F. Dressker, "On the Lifetime of Wireless Sensor Networks," *ACM Trans. Sensor Networks (TOSN)*, Vol. 5, No. 1, February 2009.

[11] A.A. Abbasi and M. Younis, "A Survey on Clustering Algorithms for Wireless Sensor Networks," *Computer Communications*, Vol. 30, No. 14, pp. 2826-2841, October 2007.

[12] R. Rajagopalan and P. K. Varshney, "Data Aggregation Techniques in Sensor Networks: A survey," *Communications Surveys & Tutorials, IEEE*, Vol. 8, No. 4, pp. 48-63, 2006.

[13] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. 33rd Hawaii Int. Conf. System Sciences (HICSS)*, Vol. 2, pp. 1-10, January 2000.

[14] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Communication*, Vol. 1, No. 4, pp. 660-670, October 2002.

[15] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid Energy-Efficient Approach," *Proc. IEEE INFOCOM*, Vol. 1, March 2004.

[16] D. Kumar, T.C. Aseri and R.B. Patel, "EECDA: Energy Efficient Clustering and Data Aggregation Protocol for Heterogeneous Wireless Sensor Networks," *Int. J. of Computers, Communications and Control*, Vol. VI, pp. 113-124, March 2011.

[17] S. Soro and W.B. Heinzelman, "Cluster Head Election Techniques for Coverage Preservation in Wireless Sensor Networks," *Ad Hoc Networks*, Vol. 7, No. 5, pp. 955-972, July 2009.

[18] B. Wang, H.B. Lim, D. Ma, and D. Yang, "A Coverage-Aware Clustering Protocol for Wireless Sensor Networks," *Proc. Mobile Ad-hoc and Sensor Networks (MSN), 2010 Sixth Int. Conf.*, Vol. 1, pp. 85-90, December 2010.

[19] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *ACM/Kluver Wireless Networks (WINET)*, Vol. 8, pp. 481-494, September 2002.

[20] X. Wang, G. Xing, Y. Zhang, and C. Lu, "Integrated Coverage and Connectivity Configuration in Wireless Networks," *ACM Transactions on Sensor Networks*, Vol. 1, No. 1, pp. 36-72, August 2005.

[21] H. Khosravi and L. Aslanyan, "SOCCP: Self Organize Coverage and Connectivity Protocol," *Proc. 2011 Third Int. Conf.*, Vol. 1, pp. 317-322, September 2011.

[22] S. Lindsey and C.S. Ranghavendra, "PEGASIS: Power-Efficient Gathering in Sensor Information Systems," *Proc. IEEE Aerospace Conf.*, Vol. 3, pp. 1125-1130, March 2002.

[23] V. Mhatre and C. Rosenberg, "Homogeneous vs Heterogeneous Clustered Networks: A Comparative Study," *Proc. IEEE ICC*, Vol. 6, pp. 3646-3651, June 2004.

[24] Z. Liu, Q. Zheng, L. Xue and X. Guan, "A distributed energy-efficient clustering algorithm with improved coverage in wireless sensor networks," *Future Generation Computer Systems*, Vol. 28, pp. 780-790, May 2012.

[25] H.O. Tan and I. Korpeoglu, "Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks," *Newsletter ACM SIGMOD Record*, Vol. 32, No. 4, December 2003.

[26] "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," *IEEE*, IEEE Standard, 802.15.4-2006, 2006.

[27] K.C. Huang, Y.S. Yen, and H.C. Chao, "Tree-Clustered Data Gathering Protocol (TCDGP) for Wireless Sensor Networks," *Proc. Int. Congo FGCN'07*, Vol. 02, pp. 31-36, 2007.

[28] S. Bandyopadhyay and E. Coyle, "An Energy-Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," *Proc. IEEE INFOCOM*, Vol. 3, pp. 1713-1723, April 2003.

[29] Y. Jin, L. Wang, Y. Kim, and X. Yang, "EEMC: An energy-efficient multi-level clustering algorithm for large-scale wireless sensor networks," *Computer Networks*, Vol. 52, pp. 542-562, February 2008.

[30] C.H. Lung and C. Zhou, "Using hierarchical agglomerative clustering in wireless sensor networks: An energy-efficient and flexible approach," *Ad Hoc networks*, Vol. 8, pp. 328-344, May 2010.

[31] M. Perillo, Z. Cheng, and W. Heinzelman, "On the Problem of Unbalanced Load Distribution in Wireless Sensor Networks," *Proc. IEEE GlobeCom Workshops*, pp. 74-79, December 2004.

[32] P.H. Hsiao, A. Hwang, H.T. Kung, and D. Vlah, "Load-Balancing Routing for Wireless Access Networks," *Proc. IEEE INFOCOM*, Vol. 2, pp. 986-995, April 2001.

[33] H. Fariborzi, and M. Moghavvemi, "EAMTR: Energy Aware Multi-Tree Routing for Wireless Sensor Networks," *IET Commun.*, Vol. 3, pp. 733-739, May 2009.

[34] F. Ren, J. Zhang, T. He, C. Lin, and S.K. Das, "EBRP: Energy-Balanced Routing Protocol for Data Gathering in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, Vol. 22, No. 12, pp. 2018-2125, December 2011.

[35] H.S. AbdelSalam and S. Olariu, "BEES: BioinspirEd BackbonE Selection in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, Vol. 23, No. 1, pp. 44-51, January 2012.

[36] Y. Tao and Y. Zhang, "Hierarchical flow balancing protocol for data aggregation in wireless sensor networks," *Proc. ComComAp*, pp. 7-12, January 2012.

[37] M. M. Hassan, B. Song, and E. Huh, "A Framework of Sensor-Cloud Integration Opportunities and Challenges," *Proc. 3rd Int. Conf. Ubiquitous Information Management and Communication (ICUIMC)*, Suwon, pp. 618-626, January 2009.

[38] N. Mitton, S. Papavassiliou, A. Puliafito, and K. S. Trivedi, "Combining Cloud and sensors in a smart city environment," *EURASIP Journal on Wireless Communications and Networking*, Vol. 2012, No. 1, pp. 247-256, 2012.

[39] N. Poolsappasit, V. Kumar, S. Madria, and S. Chellappan, "Challenges in Secure Sensor-Cloud Computing," *Lecture Notes in Computer Science*, Vol. 6933, pp. 70-84, 2011.

[40] H. Zhang and J. C. Hou. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. *Ad Hoc Sensor Wireless Networks*, Vol. 1, pp. 89-124, March 2005.

[41] R. Panta, S. Bagchi, I. Khalil, and L. Montestruque. Single versus multi-hop wireless reprogramming in sensor networks. *Proc. 4th Int. Conf. TridentCom*, pp. 1-7, February 2008.

[42] F. Librino, M. Levorato, and M. Zorzi, "An Algorithmic Solution for Computing Circle Intersection Areas and Its Applications to Wireless Communications," *Proc. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2009. 7th Int. Symp.*, Vol. 1, pp. 1-10, June 2009.

[43] M. Bhardwaj and A. Chandrakasan, "Upper Bounds on the Lifetime of Wireless Sensor Networks" *Proc. IEEE ICC*, Vol. 3, pp. 785-790, June 2001.

[44] P. Basu and J. Redi, "Effect of Overhearing Transmissions on Energy Efficiency in Dense Sensor Networks," *Proc. 3rd Int. Sym. IPSN '04*, pp. 196-204, April 2004.

[45] F. Chen, X. Tong, E. Ngai, and F. Dressler, "Mode Switch - Adaptive Use of Delay-Sensitive Or Energy-Aware Communication in IEEE 802.15.4-Based Networks," *Proc. 7th IEEE Int. Conf. MASS*, pp. 302-311, November 2010.

[46] M. Kurz, G. Holzl, and A. Ferscha, "Goal-Oriented Opportunistic Sensor Clouds," *Lecture Notes in Computer Science*, Vol. 7566, pp. 602-619, 2012.

[47] Y. Wang, J, Wu, S. Tang, *et al*, "A Survey of Virtual Machine Placement in Cloud Computing for Big Data", *CRC Press, Taylor & Francis Group*, Roca Raton, FL, 2016.

[48] R. Shah, "Reduce Network Traffic with Web Caching", *IBM developerWorks*, Sept. 1, 1999

[49] W. Liao, K. Coloma, A. Choudhary, L. Ward, E. Russell, and S. Tideman, "Collective caching: Application-aware client-side file caching," *Proc. 14th IEEE Int. Symposium on High Performance Distributed Computing (HPDC)*, pp. 81-90, 2005.

[50] S. Romano and H. ElAarag, "A Quantative Study of Web Cache Replacement Strategies using Simulation," *Simulation*, Vol. 88, No. 5, pp. 507-541, 2011.

[51] C. Ge, Z. Sun, and N. Wang, "A Survey of Power-Saving Techniques on Data Centers and Content Delivery Networks," *IEEE Communications Surveys and Tutorials*, Vol. 15, No. 3, pp. 1334-1354, 2013.

[52] B. Tan and L. Massoulie, "Optimal content placement for peer-to-peer video-on-demand systems," *IEEE/ACM Transactions on Networking*, pp. Vol. 21, No. 2, pp. 566-579, September 2011.

[53] M. Bjorkqvist, L. Y. Chen, M. Vukolic, and X. Zhang, "Minimizing Retrieval Latency for Content Cloud," *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, Shanghai, China, pp. 1080-1088, April 2011.

[54] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa, "Supporting Cooperative Caching in Disruption Tolerant Networks," *Proc. 31st International Conference on Distributed Computing Systems (ICDCS)*, Minneapolis, pp. 151-161, June 2011.

[55] R. Z. Farahani and M. Hekmatfar, "Facility Location: Concepts, Models, Algorithms and Case Studies," *Springer-Verlag, Berlin Heidelberg*, New York, pp. 177-242, 2009.

[56] Z. Drezner and H. W. Hamacher, "Facility Location: Applications and Theory," *Springer-Verlag Berlin Heidelberg*,New York, pp. 37-202, 2001.

[57] M. S. Daskin, "Network and Discrete Location: Models, Algorithms, and Applications," *John Wiley and Sons*, New York, pp. 247-303, 1995.

[58] http://www.noaa.gov.

[59] M. L. Fisher, "An Applications Oriented Guide to Lagrangian Relaxation," *Interfaces*, Vol. 15, No. 2, pp. 10-21, April 1985.

[60] Amazon Simple Storage Service, http://aws.amazon.com/s3/pricing/.

[61] , J. Wang, X. Yang, Z. Zhang, B. Li, and J. Kim, "A Survey about Routing Protocols with Mobile Sinks for Wireless Sensor Network," *International Journal of Future Generation Communication and Networking*, Vol. 7, No. 5, PP. 221-228, 2014.

[62] Y. Gu, Y. Ji, J. Li, and B. Zhao, "ESWC: Efficient Scheduling for the Mobile Sink in Wireless Sensor Networks with Delay Constraint," *IEEE Transaction on Parallel and Distributed Systems*, Vol. 24, No. 7, July 2013.

[63] J. Wang, Y. Yin, J. Kim, S. Lee, and C. Lai, "An Mobile-sink Based Energy-efficient Clustering Algorithm for Wireless Sensor Networks," *IEEE International Conference on Computer and Information Technology*, Chengdu, China, pp. 678 - 683, October 2012.

[64] W. Aioffi, G. Mateus, and F. Quintao, "Optimization Issues and Algorithms for Wireless Sensor Networks with Mobile Sink". *International Network Optimization Conference*, Spa, Belgium, April 2007.

[65] K. Kumar and Y. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? " *Computer*, Vol.43, No.4, pp. 51-56, April 2010.

[66] K. Kumar, J. Liu, Y. Lu, and B. Nhargava, "A Survey of Computation Offloading for Mobile Systems," *Mobile Networks and Applications*, Vol. 18, No. 1, pp. 129-140, April 2012.

[67] M. Bolat, K. Kelsey, X. Li, and G. Gao, "Source Code Partitioning in Program Optimization," *IEEE 17th International Conf. on Parallel and Distributed Systems (ICPADS)*, pp. 56-63, Tainan, December 2011.

[68] R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, "Wishbone: Profile-based Partitioning for Sensornet Applications," *6th USENIX Symposium on Networked Systems Design and Implementation*, Vol. 9, pp. 395-408, April 2009.

[69] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 40, No. 4, pp. 23-32, March 2013.

[70] Y. Ge, U. Zhang, Q. Qiu, and Y. Lu, "A Game Theoretic Resource Allocation for Overall Energy Minimization in Mobile Cloud Computing System," *Proc. ISLPED'12*, pp. 279-284, California, July 2012.

[71] Y. Tao, Y. Zhang, and Y. Ji, "Efficient Computation Offloading Strategies for Mobile Cloud Computing," *Advanced Information Networking and Applications*, Gwangju, Korea, pp. 626-633, March 2015.

[72] I. Khan, N. Jabeur, M. Khan, and H. Mokhtar, "An Overview of the Impact of Wireless Sensor Networks in Medical Health Care," *Int. Conf. on Computers and Information Technology*, Hammamet, Tunisia, May 2012.

[73] V. Shnayder, B. Chen, K. Lorincz, T. FulfordJones, and M. Welsh, "Sensor Networks for Medical Care," *SenSys*, Vol. 5, pp. 314-314, 2005.

[74] M. Afsaneh, A. Ali, S. Paymon, and S. Reza, "Application of Wireless Sensor Networks in HealthCare System," *Proc. ASEE Conference and Exposition*, Atlanta, Georgia, June 23-26, 2013.

[75] M. Kafi, Y. Challal, D. Djenouri, M. Doudou, A. Bouabdallah, and N. Badache, "A study of wireless sensor networks for urban traffic monitoring: applications and architectures," *Procedia Computer Science* , Vol. 2013, No. 19, pp. 617-626, 2013.

[76] K. Yousef, M. Al-Karaki, and A. Shatnawi, "Intelligent Traffic Light Flow Control System Using Wireless Sensors Networks," *Journal of Information Science and Engineering*, Vol. 26 No. 3, pp. 753-768, 2010.

[77] V. Gungor, B. Lu, and G. Hancke, "Opportunities and Challenges of Wireless Sensor Networks in Smart Grid," *IEEE Transactions on Industrial Electronics*, Vol. 57, No. 10, October 2010.

[78] Y. Lim, H. Kim, and S. Kang, "A design of wireless sensor networks for a power quality monitoring system," *Sensors*, pp. 9712-9725, 2010.

Publication list

1) Yaling Tao and Yongbing Zhang, “Hierarchical flow balancing protocol for data aggregation in wireless sensor networks”, Computing, Communications & Applications (ComComAp) Conference, pp. 7-12, Hong Kong, China, January 2012 (Chapter 3)

2) Yaling Tao, Yongbing Zhang, and Yusheng Ji, “Flow-Balanced Routing for Multi-Hop Clustered Wireless Sensor Networks”, Ad hoc networks, Elsevier B.V. , Vol. 11, No. 1, pp. 541-554, 2013 (Chapter 3)

3) Yaling Tao, Yongbing Zhang, and Yusheng Ji, “Optimal Data Cache Allocation for Mobile Devices in Sensor-Cloud”, IPSJ SIG Mobile Computing and Ubiquitous Communications (MBL), 2013-MBL-69(13), pp. 1-8, Saitama, Japan, December 2013. (Chapter 4)

4) Yaling Tao, Yongbing Zhang, and Yusheng Ji, “Efficient Data Cache Allocation for Sensor-Clouds”
(IET Communications, under the second review) (Chapter 4)

5) Yaling Tao, Yongbing Zhang, and Yusheng Ji, “Efficient Computation Offloading Strategies for Mobile Cloud Computing”, Advanced Information Networking and Applications (AINA-2015), Gwangju, Korea, March 2015.