Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 296296, 14 pages
doi:10.1155/2012/296296

Research Article

An Efficient Data-Gathering Scheme for Heterogeneous Sensor Networks via Mobile Sinks

Po-Liang Lin and Ren-Song Ko

Department of Computer Science and Information Engineering, National Chung Cheng University, 168 University Road, Min-Hsiung Chia-Yi 621, Taiwan

Correspondence should be addressed to Ren-Song Ko, korenson@cs.ccu.edu.tw

Received 16 June 2011; Revised 27 August 2011; Accepted 27 August 2011

Academic Editor: Yuhang Yang

Typical Wireless Sensor Networks (WSNs) use static sinks to collect data from all sensor nodes via multihop forwarding. This results in the hot spot problem since the nodes close to the sink have a tendency to consume more energy in relaying data from other nodes. Many approaches using mobile sinks have been proposed to prevent this problem, but these approaches still suffer from the buffer overflow problem due to the limited memory capacity of the sensor nodes. This paper proposes an approach in which the mobile sink traverses a subset of nodes. Given the characteristics of wireless communication, such an approach can effectively alleviate the buffer overflow problem without incurring additional energy consumption. To further alleviate the buffer overflow problem, we propose the *Allotment Mechanism* which allows nodes with different data sampling rates to share their memory and, thus, extend the overflow deadline. Finally, the effectiveness of the proposed approach is verified via the GloMoSim network simulator. The results show that our approach incurs fewer buffer overflows than other data-gathering schemes.

1. Introduction

Wireless Sensor Networks (WSNs) play an important role in a wide range of applications [1, 2], such as routine data collection [3, 4], distant unfriendly location exploration [5, 6], emergency response [7, 8], and hazard detection [9]. A WSN may consist of hundreds or thousands of sensor nodes that integrate sensors with limited onboard processing and wireless communication capabilities. With the available technology, the sensors are usually battery powered and have limited memory. It is not economically or technically feasible to recharge or replenish their energy. In addition, hardware constraints limit the amount of sensed data that can be stored in a sensor node. Data may, therefore, need to be discarded if it cannot be processed before the buffer overflows [10]. Therefore, buffer overflow and network lifetime are often seen as the two most important issues in designing a WSN.

Typical WSNs have static data sinks for data gathering, in which sensor nodes transmit data to sinks via multihop forwarding [11–13]. In other words, a node not only transmits the information sensed by itself but also relays the data packets generated by others. Therefore, nodes near sinks have a tendency to consume more energy because they may need to relay data for nodes at a distance from sinks. Multihop forwarding may create a hot spot problem which may exhaust the nodes near sinks, leaving the sinks isolated from the rest of the network, and the networks will cease to function since the data sensed by the remote nodes cannot be forwarded to the sinks, thus, limiting the utility of the remaining working nodes.

Many studies have proposed the use of mobile sinks for sensor networks [14, 15] to solve the hot spot problem and improve energy efficiency in data gathering. Instead of waiting for data, a mobile sink (MS) can travel around the network, approach individual nodes, and collect data from them. For example, in [16], an MS can randomly roam around the region of interest (ROI) and collect data from sensor nodes. The authors of [17, 18] considered the mobility planning problem which involves determining a good traverse path or rendezvous point for the MS to optimize the energy consumption used in communicating with each sensor node. Note that, intuitively, the MS can approach

each node to avoid the need for multihop forwarding, and thus minimize communication energy consumption. The limited memory size of sensor nodes means a node can hold or buffer data before memory is full, and a node may be unable to hold its generated data in memory for collection by the MS if the MS traverses each node inappropriately. As a consequence, such a buffer overflow problem may result in information loss. Determining a traverse path within a given deadline is known as the *Traveling Salesman Problem* (TSP) and its complexity is NP complete. Rather than searching for a moving path without any buffer overflow, Somasundara et al. [32] proposed a mobile element scheduling method to collect data with dynamic deadlines with the goal of minimizing the number of missed deadlines. They used the moving cost of the MS and the buffer overflow time of the sensor nodes to determine the visiting sequence of sensor nodes. Nevertheless, this scheduling method requires the MS to visit all sensor nodes. A large number of nodes spread over the ROI would result in a long moving path, reducing MS efficiency, and some nodes may still suffer the buffer overflow problem. Furthermore, once a buffer overflow occurs, the data is simply dropped. Salvaging the data is not considered.

This paper tries to address the energy consumption problem of data gathering and avoid the buffer overflow problem by using a hybrid approach that combines one-hop data forwarding and MS. Note that traditional static sink approach has the hot spot problem mentioned above, but MS has the minimal buffer overflow problem. On the other hand, using MS to traverse each node alleviates the hot spot problem at the cost of incurring buffer overflow. By combining data forwarding and MS, we can designate several rendezvous points for the MS to visit. Once the MS reaches a rendezvous point, the sensor nodes close to the rendezvous point will send their buffered data to the MS via data forwarding. Increasing the number of rendezvous points decreases the severity of the hot spot problem but increases the severity of the buffer overflow problem. That is the number of hops for data forwarding and the number of rendezvous points are a tradeoff between the hot spot problem and the buffer overflow problem and, thus, a key challenge to data gathering. In this paper, the maximal number of hops to the closest rendezvous point is restricted to one, and our proposed approach is then reduced to determine an appropriate path to traverse these rendezvous points within the data overflow deadlines. The WSN application considered here is environmental monitoring or surveillance with heterogeneous sensor nodes, in which the nodes may have different hardware, sensing capability, or data sampling rates. The proposed approach requires an MS, a robot, or a vehicle equipped with a transceiver to collect the data from the nodes that are within its one-hop communication range. Besides, a node will buffer the generated data in its memory if there is no MS within its one-hop communication range. The objective is to have the MS to collect the data from nodes before their buffers are full. We achieve this objective by minimizing the rendezvous points and determining a good traverse path.

Our proposed approach for efficient data gathering can be summarized as follows.

(1) We cluster the sensor nodes by constructing a dominating set [20, 21] and assign the dominating nodes as rendezvous points. By definition, a node is either a dominating node or within a one-hop communication range of its closest dominating node. After that, the MS only needs to traverse the dominating nodes for data gathering. There is no need for dominating nodes to buffer the data of its one-hop neighbors. When MS arrives at a dominating node, it will collect the data from the dominating node and its one-hop neighbors via one-hop data transmission.

(2) We also propose an *Allotment Mechanism* to alleviate the buffer overflow problem, specifically for sensor nodes with high sampling rates. The *Allotment Mechanism* will average the amount of sensed data in the cluster. Nodes with higher sampling rates may temporarily buffer their data in the memory of the node with a lower sampling rate in the same cluster. Doing so can extend the time to buffer overflow for the nodes with the higher sampling rates, and the overall buffer overflow problem is alleviated.

After determining rendezvous points, we consider two algorithms for the rendezvous point traversing problem with deadline constraints, namely the *Dominating-Based Minimum Weighted Sum First* (DMWSF) *algorithm* and the *Dominating-Based Traveling Salesman Approximation* (DTSP) *algorithm*. The objective is to minimize the number of missed deadlines for a given data sampling rate, or the dual problem, to maximize the data sampling rates without any missed deadlines.

To verify the effectiveness of the proposed approach, we conducted simulations with the GloMoSim simulator [22] and compared the results with those from other related algorithms. The results illustrate that the proposed approach significantly outperforms other algorithms in terms of providing longer network lifetime and less buffer overflow.

The rest of this paper is organized as follows. Section 2 discusses the related work with regard to data-gathering schemes via MS. The sensor network model, assumptions, and notations are introduced in Section 3, and the proposed algorithms are described in Section 4. Section 5 describes and discusses the simulation environment, simulation parameters, and simulation results. Conclusions are given in the last section.

2. Related Work

As mentioned earlier, MS data-gathering schemes may avoid the hot spot problem and prolong the network lifetime. Many approaches have been proposed for WSN [19, 23, 24], and one major design challenge is to plan a traverse path without missing any deadlines. One intuitive solution is to traverse each sensor node in the shortest distance so that the network can tolerate buffer overflow problems with a high sampling rate [25]. Wohlers et al. [26] pointed out that whether a data-gathering approach is energy efficient depends on the application characteristics, including the

mobility patterns of sinks and the desired freshness of the collected data.

Several researchers [27, 28] have proposed using either a tree, cluster, or grid structure to level down the large-scale sensor network. In [29] an energy-efficient routing protocol Multitier Grid Routing Protocol (MGRP) is proposed which introduces a special hybrid multi-tier structure for data dissemination. They form an optimized cluster which transmits reliable data to its higher tier cluster head, with the uppermost cluster head from neighbor grids further negotiating to construct the data d-tree from which the mobile sink can access and send queries. After reducing the solution space, some proposed methods construct the optimal traverse path for mobile sink. In [30] this problem is addressed by minimizing the traverse length of the mobile elements (MEs), seeking to obtain optimal scheduling for the MEs based on the minimal stop point set to minimize their traverse distance.

In [16, 31], Shah et al. proposed a data-gathering scheme using a mobile sink called Mobile Ubiquitous LAN Extensions (MULEs) which is basically a moving observer, such as an animal or human being, carrying a transceiver. The mobile observer wanders in the ROI, collecting and transmitting information to access points for further processing and analysis. For a performance comparison with our proposed approach, this approach is also implemented in the GloMoSim simulator and denoted as Random_Waypoint.

As mentioned above, one possible traverse path for alleviating the buffer overflow problem is the one with the shortest distance, a TSP with NP-complete complexity. Therefore, instead of permuting the node visiting order to find the minimal traverse path, Somasundara et al. [32] proposed an approach called k-lookahead in which only k nodes (k is less than 10) are permuted at a time and the first k node visited is the one with the minimum cost among k nodes. Furthermore, they also proposed a mobile element scheduling method to collect data in [32]. The MS calculates a weighted value and uses the Minimum Weighted Sum Value First (MWSF) algorithm to determine the next destination. The weighted value consists of two factors, cost and deadline, in which the cost is the distance between the MS and the potential destination node and the deadline is the time to buffer overflow for the potential destination node. If the weighted value is dominated by the cost, the MS will traverse the nodes by the shortest distance; if dominated by the deadline, the MS will visit the node with the earliest deadline first. This algorithm is also called the Early Deadline First (EDF) algorithm in [32]. The performance results in [32] show that the buffer overflow ratio of MWSF is similar to k-lookahead for $k = 6$, while MWSF is computationally inexpensive.

Ma and Yang [17] introduced an energy-efficient data-gathering scheme with an MS called SenCar. The fundamental idea is to determine an energy-efficient traverse path via a bisection method. Given starting and ending points, the MS can move straight across the ROI to gather the data from sensor nodes via multihop forwarding. While a straight line may not be an energy-efficient moving path, it is possible to find a turning point in the middle. As a consequence, the moving path moves from the starting point to the turning point and then to the ending point, and the MS will gather the data along the path using less energy. Furthermore, more new turning points can be recursively added between the starting point, turning points, and the ending point for better energy efficiency. Note that, though more turning points may reduce the number of hops for data forwarding, which in turn reduces energy consumption, it will increase the traverse distance and the MS will need more time to cross the ROI, which may produce the buffer overflow problem. We also include this approach, denoted as BISECTOR, in our simulation for performance comparison.

Xing et al. [18] proposed a rendezvous-based data-gathering scheme. A subset of nodes is designated as rendezvous points that will buffer and aggregate data originating from other sensor nodes. The MS will traverse each rendezvous point to collect these data. Conceptually, these rendezvous points serve as temporary static sinks from which the MS collects data; thus, this approach basically distributes the hot spot problem over these rendezvous points. Rao and Biswas [33] introduced a distributed ant-based TSP mechanism to determine a traverse path for MS such that all the chosen rendezvous point are visited.

Saad et al. [34] proposed a hierarchical structure for large-scale sensor networks via a clustering algorithm. Cluster heads are randomly selected in time-driven scenarios, and the MS will traverse these cluster heads to minimize the energy consumption on multihop forwarding. However, this approach does not consider the buffer overflow problem.

This paper considers heterogeneous WSNs, in which sensor nodes may have different data sampling rates and, thus, different levels of severity of the buffer overflow problem. To verify the effectiveness of our proposed approach, several other approaches, namely, Random_Waypoint, BISECTOR and MWSF, are implemented in the simulation for performance comparison for factors including network lifetime and number of buffer overflows.

3. Network Model and Assumptions

In this paper, a heterogeneous WSN with different data sampling rates can be modeled as follows.

(1) A connected graph consists of n nodes with unique ID $1, \ldots, n$ and an MS that moves around the ROI.

(2) Sensing operations of sensor nodes, such as temperature and humidity, are determined prior to deployment and set with given sampling rates (which may be different). The vector $S[1, \ldots, n]$ denotes the sampling rates (i.e., bytes per unit of time) of the sensor nodes.

(3) Each sensor node has a limited memory size (which may be different) represented by the vector $M[1, \ldots, n]$.

(4) All sensor nodes and the MS have the same communication range, denoted as R_C.

(5) The MS is aware of the position of each sensor node.

Table 1: The notation list.

Description	Notation
Overflow time	T_o
Maximum overflow time	$T_{o\_\max}$
Sample rate	S
Memory size	M
Election timer	T_{elk}
Maximum election timer	$T_{\text{elk}\_\max}$
Timer to overflow time ratio	η
Overflow time with *Allotment Mechanism*	D_o
Allotment data	AD

(6) The MS is initially stopped, and all nodes know how to send data to the MS (In this case, the MS acts like a static sink.) In addition, once the MS begins moving, it does so with a constant speed, v m per unit of time.

In addition to the WSN model above, the following assumptions are made.

(1) Any two nodes can directly communicate via bi-directional wireless links if their Euclidean distance is not greater than R_C.

(2) All nodes have their clocks synchronized.

(3) The actual data transfer time from a node to the MS is negligible when calculating the buffer overflow time.

Based on the model and assumptions above, it is easy for the MS to derive the following information.

(1) Since the MS knows the position of each node, it can derive the distance for each pair of nodes, denoted as D_{ij} for node i and j. Furthermore, the matrix $\text{cost}[1,\ldots,n][1,\ldots,n]$ that denotes the time needed for the MS to move from one node to another is defined as

$$\text{cost}[i][j] = \frac{D_{ij}}{v}. \tag{1}$$

(2) The buffer overflow time or the time to fill the memory of node i, denoted as $T_o[i]$, is defined as

$$T_o[i] = \frac{M[i]}{S[i]}. \tag{2}$$

Table 1 lists the notations used in this paper.

Given the model and assumptions above, this paper addresses two research issues:

(1) reducing the number of hops from sensor nodes to the MS, which in turn will reduce the energy consumption of sensor nodes and

(2) determining a traverse path for MS to alleviate the buffer overflow problem, that is, minimizing the number of missed buffer deadlines.

The proposed approaches are described in detail in the next section.

4. Algorithms

4.1. Overview. One intuitive way to reduce communication energy consumption is to have the MS approach in each node to collect data. However, due to the NP completeness of TSP, it is infeasible to both traverse each node and meet the buffer overflow deadline, particularly in large-scale networks. Note that, in WSN, sensor nodes communicate with each other via wireless communication. As long as the MS is in the communication range of a sensor node, it can collect the sensor node's data. Therefore, instead of visiting each node, we designate some sensor nodes as rendezvous points so that each node is located within a one-hop communication range of at least one of these rendezvous points. By only traversing these rendezvous points, the MS can collect all data from the sensor nodes via wireless communication without expending additional energy on communication. One possible set of such rendezvous points is the dominating set of WSN in which, by definition, each node is either a dominating node or within a one-hop range of a dominating node [35, 36]. With fewer nodes to visit, it is easier to plan the MS's traverse path and meet the buffer overflow deadlines.

Moreover, sensor nodes may have different sampling rates and, thus, different buffer overflow deadlines. To further extend the deadlines of nodes with higher sampling rates and, thus, alleviate the buffer overflow problem, we allow the nodes with higher sampling rates to temporarily buffer their data in the memory of their one-hop neighbors that have lower sampling rates. Basically, our proposed approach consists of three modules.

(1) The first module reduces the scale of the WSN so that the MS may collect all data without traversing all nodes and without sensor nodes needing to relay data from other nodes. This is achieved by letting each sensor node run the *Time Delay-Based Dominating* (TDD) *algorithm* and use its own buffer overflow time to select dominating nodes.

(2) Sometimes analyzing variations in the ROI may require maintaining the completeness of data collected by sensor nodes. Therefore, the *Allotment Mechanism* is proposed here to alleviate the buffer overflow problem by letting nodes temporarily buffer their data in the memory of their one-hop neighbors.

(3) The last module determines the MS's traverse path so that missed buffer overflow deadlines can be avoided or minimized. Note that, though the problem scale is reduced to the dominating set, the complexity is still NP complete. Therefore, instead of finding the optimal path, this paper considers a heuristic algorithm and an approximation algorithm, namely the *Dominating-Based Minimum Weighted Sum First* (DMWSF) *algorithm* and the *Dominating-Based Traveling Salesman Approximation* (DTSP) *algorithm* respectively.

Each module will be described in detail in the following subsections.

4.2. Time Delay-Based Dominating (TDD) Algorithm . Some rendezvous-based data-gathering schemes, such as [18, 37], utilize rendezvous points as static sinks. The data collected by other nodes will be forwarded to these rendezvous points via multihop forwarding for the MS to collect. Thus, the hot spot problem still exists and is merely transferred from a centralized data sink to these rendezvous points. For example, some rendezvous-based schemes ask the rendezvous points to buffer data originated from other nodes and then forward the buffered data to the MS when it arrives. Such an approach raises both the hot spot problem and the buffer overflow problem due to the limited memory size of the rendezvous points. Note that the main cause of the hot spot problem is the multihop forwarding in which the nodes closest to the sinks are more likely to relay data from other nodes and, thus, consume more energy for communication. Thus, we consider using the dominating set as the rendezvous points, not only to decrease the scale of TSP but also to eliminate the hot spot problem.

Definition 1. A dominating set of a graph $G = (V, E)$ is a subset $S \subseteq V$ of the nodes such that, for all nodes $u_1 \in V$, either $u_1 \in S$ or a neighbor u_2 of u_1 is in S. Here, if the distance between u_2 and u_1 is less than R_C, u_2 and u_1 are neighbors. Besides,

(i) $u_1 \in V$ is the dominating header (*DH*) if $u_1 \in S$;

(ii) $u_1 \in V$ is the dominating member (*DM*) if u_1 is a neighbor of $u_2 \in S$;

(iii) a *DH* and its *DM* form a dominating cluster (*DC*).

Referring to Algorithm 1, we propose a time-based algorithm TDD to determine *DH*, *DM*, and *DC* based on sensor node's timers, T_{elk}. The main purpose of this algorithm is to reduce information exchange and to construct *DC* in a fixed amount of time denoted as $T_{\text{elk\_max}}$. To integrate this with *Allotment Mechanism*, described in the next subsection, we use buffer overflow time to define timers in dominating header selection; that is,

$$T_{\text{elk}}[i] = \eta \times T_o[i]. \tag{3}$$

To guarantee that TDD will terminate in $T_{\text{elk\_max}}$, η can be determined by

$$\eta = \frac{T_{\text{elk\_max}}}{T_{o\_\max}}. \tag{4}$$

All the notations mentioned here are listed in Table 1. Note that the parameters η and $T_{o\_\max}$ are prerequisites for TDD and need to be given to each sensor node in advance. After deploying the sensor nodes in the ROI, each sensor node timer starts to countdown. If a sensor node does not receive any *DH* declaration message from its neighbors before its election timer expires, it will declare itself to be *DH* and broadcast a *DH* declaration message to its neighbors. The *DH* declaration message contains the buffer overflow time of the *DH*. Thus, if a sensor receives more than one *DH* declaration, it may select the *DH* with the least buffer overflow time or resort to some tie-breaking mechanism if the buffer overflow times are same. On the other hand, if a sensor node receives a *DH* message from its neighbors, it will declare itself to be *DM* and reply to its *DH* neighbor with a *DM* declaration message. The TDD algorithm is presented in Algorithm 1.

```
(1) set timer = T_elk[i]
(2) while timer ≠ 0 do
(3)     timer − −
(4) end while
(5) if no DH declaration message from its neighbors then
(6)     declare as DH and broadcast a DH declaration
        message to its neighbors
(7) else
(8)     declare as DM and reply to its DH neighbor with a
        DM declaration message
(9) end if
```

Algorithm 1: Time delay-based dominating algorithm.

Note that the MS is initially stopped and all nodes know how to send data to the MS. After the *DH*s are selected, they may notify the MS with all their *DM*s. In addition, the MS has knowledge about all nodes and, thus, knows the TDD results of all sensors in maximum election time, $T_{\text{elk\_max}}$. The MS will then start to traverse *DM*s to collect data. To cope with packet loss due to unreliable communication conditions, it is possible to assign the MS a timer with the expiration time greater than $T_{\text{elk\_max}}$. Once expired, the MS begins to traverse the *DH*s it knows regardless of information incompleteness. Besides, the MS will visit the nodes for which TDD results are unknown to the MS to collect their TDD results. In other words, it is still possible for the MS to know all the *DH*s.

From the definition of the election timer, a *DH* obtained from TDD will have a shorter buffer overflow time than its neighbors; that is, the *DH* has the critical buffer overflow time of its *DC*. Thus, the buffer overflow time for a *DC* can be represented by the buffer overflow time of its *DH*, and the MS only needs to consult each *DH*'s buffer overflow time to plan a traverse path which minimizes the number of missed buffer deadlines. As long as the MS moves to each *DH* before its buffer overflow deadline, it can collect the data from the *DH* and all its *DM*s without any data loss.

Note that if a *DH*'s *DH* declaration message is delayed or lost during the setup of the dominating clusters, its neighbors may declare themselves to be *DH*s after election timers expire, thus, increasing the number of clusters and the complexity of the traverse path planning. However, this problem may be alleviated by pruning the dominating clusters, for example, by modifying the approaches proposed in [35] in which we may use buffer overflow time instead of node ID to determine which cluster will be pruned. On the other hand, when a *DM* declaration message is delayed or lost, a *DH* may not recognize its neighbor as a *DM*, which may cause the *Allotment Mechanism* described later fail since it does not have the complete information of its neighbors' data-sampling rates. This problem may be recovered when

the MS visits the *DH*; all its *DMs* have a chance to update their membership while sending data to the MS.

4.3. Allotment Mechanism. Note that the buffer overflow deadline of a *DC* is determined by the deadline of the node with the highest sampling rate, that is, the *DH*. To further extend the *DH*'s deadline and, thus, alleviate the buffer overflow problem, we introduce the *Allotment Mechanism* which allows a *DH* to temporarily buffer its data in the memory of its *DMs*. The basic idea is to select the first k highest sampling rate nodes to share their memory with the *DH*.

For the purposes of discussion, we sort m member nodes of a given *DC* in descending order of data sampling rates and relabel them as $u_0, u_1, \ldots, u_m$; that is, the sampling rate of u_i is not lower than that of u_{i+1}. With the *Allotment Mechanism*, the *DH*, that is, u_0, may distribute its data to $u_1, u_2, \ldots, u_k$, so its buffer will not be filled so quickly, thus, extending its buffer overflow time, along with the *DC*.

Note that the deadline of a *DC* is determined by the deadline of its *DH*, for example, node u_0, which is $T_o[u_0]$ defined in (2). With the *Allotment Mechanism*, the buffer overflow time of the *DH* or *DC* is no longer $T_o[u_0]$ and is extended to $D_o[u_0]$, defined as

$$D_o[u_0] = \frac{\sum_{j=0}^{k} M\left[u_j\right]}{\sum_{j=0}^{k} S\left[u_j\right]}. \tag{5}$$

Referring to Table 1, M is the memory size of the sensor nodes. Note that (5) is derived from the fact that the total memory of the *DH*, and k *DMs* is shared, and, thus, the overflow deadline is the total memory size divided by total sampling rates of *DH* and k *DMs*.

Consider the example depicted in Figure 1. u_0 is *DH* and has three *DMs*, u_1, u_2, and u_3, in the same *DC*. The order of the *DMs*' sampling rates is $u_0 > u_1 > u_2 > u_3$. If we share the memory of 2 *DMs* in the *Allotment Mechanism*, the memory of node u_1 and u_2 will be shared. Note that u_0 has the highest sampling rate in the *DC*, followed by u_1 and u_2. If the memory of u_0 is almost filled, it may free some of its memory space by temporarily storing some of its data into u_1's or u_2's memory. Thus, the space released in u_0 can be used to store more data and extend the buffer overflow deadline.

A *DH* can easily determine the order of its *DMs*' sampling rates, since each *DM* will return a *DH* with the *DM* declaration message once the election timer expires. The election timer is proportional to the buffer overflow time, and, thus, the order of the *DM* declaration message mirrors the order of the sampling rates. Therefore, the first k *DMs* sending the *DM* declaration messages will share memory with the *DH*.

Basically, u_0 calculates the quantity of allotment data of the k members, $u_1, u_2, \ldots, u_k$, by the following:

$$\mathrm{AD}\left[u_j\right] = \left(T_o\left[u_j\right] - D_o[u_0]\right) \times S\left[u_j\right], \quad 1 \le j \le k. \tag{6}$$

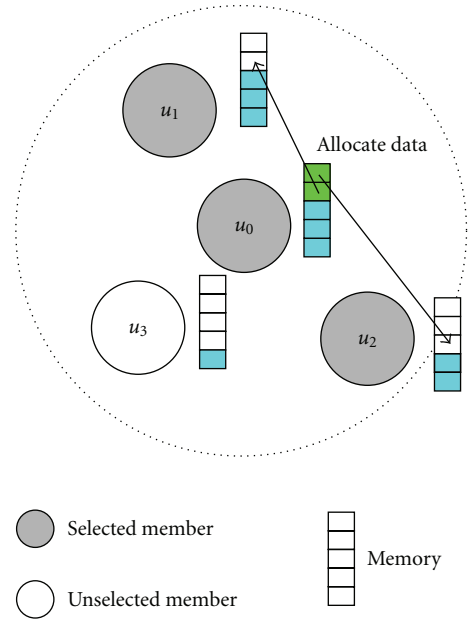


Figure 1: An example of the *Allotment Mechanism*. The *DH*, u_0, of a *DC* may distribute its data to some *DMs*, u_1 and u_2, in the same *DC* to free some of its memory space.

Thus, if the $\mathrm{AD}[u_j]$ is positive, u_j can temporarily save $\mathrm{AD}[u_j]$ bytes data for other sensor nodes; on the other hand, if the $\mathrm{AD}[u_j]$ is negative, u_j needs to forward the $\mathrm{AD}[u_j]$ bytes data to other sensor nodes. Which nodes will cooperate with u_j for sharing memory is decided by node u_0 through the allotment algorithm, a flowchart of which is presented in Figure 2. Note that the *DH* in each dominating set will first select argument k to calculate the AD array and then execute the allotment algorithm.

The following theorem says no *DM* of u_0 has a buffer overflow time less than $D_o[u_0]$. Thus, we can use $D_o[u_0]$ to represent the buffer overflow time of a *DC*. That is, the *Allotment Mechanism* can more evenly distribute the data to the memory of nodes in a heterogeneous WSN.

Theorem 2. *For a given DC, the buffer overflow time of u_j, $j \neq 0$, is not less than the buffer overflow time of u_0.*

Proof. If u_j is not selected to share its memory; that is, $k < j \le m$, we have

$$\frac{M[0]}{S[0]} \le \cdots \le \frac{M[u_k]}{S[u_k]} \le \frac{M\left[u_j\right]}{S\left[u_j\right]}. \tag{7}$$

Therefore,

$$D_o[u_0] = \frac{\sum_{i=0}^{k} M[u_i]}{\sum_{i=0}^{k} S[u_i]} \le \frac{M\left[u_j\right]}{S\left[u_j\right]} = T_o\left[u_j\right]. \tag{8}$$

On the other hand, the proof is trivial if u_j is designated to share its memory since its buffer overflow time is $D_o[u_0]$. □

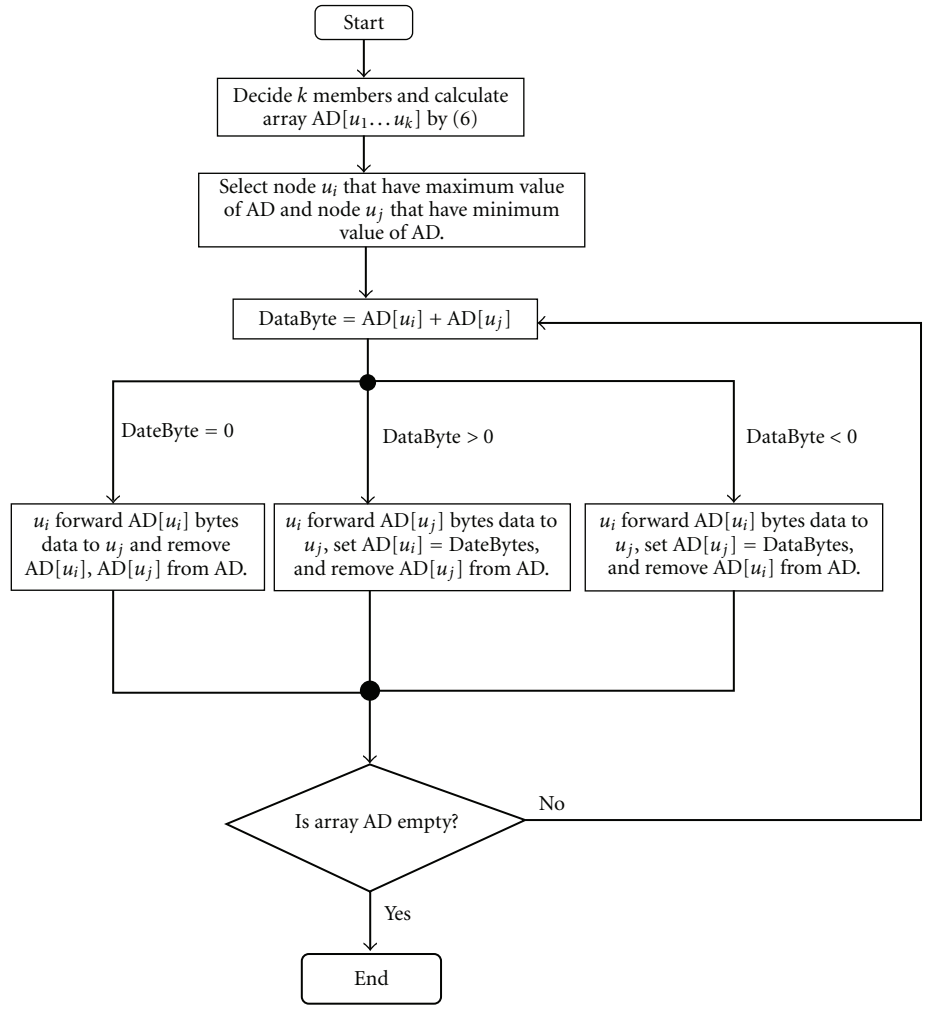


Figure 2: Flowchart of the allotment algorithm.

Note that the reason that the proposed *Allotment Mechanism* does not let a *DH* (i.e., u_0) store data on *DMs* with lower sampling rates is that the buffers of these sensors sharing memory cannot be filled slower than the buffer of the sensor having the second highest sampling rate, that is, u_1. Otherwise, the most critical node of the *DC* becomes u_1, not u_0. Therefore, it is necessary to take u_1 into consideration. However, if both u_0 and u_1 store data on *DMs* with the lower sampling rates, it is necessary to consider u_2 for the same reason. Thus, we simply have sensors with higher sampling rates share their buffers to minimize management overhead.

The larger k allows more *DMs* to share their memory at the cost of greater data management and communication overhead, for example, to determine which node has free memory to share and transmit the data via wireless communication. As a result, the *DH* may consume more energy on computation and communication. Therefore, due to the sensor nodes' limited communication and computation resources, k cannot be too large in practical applications. With the *Allotment Mechanism*, every *DC* has a larger buffer overflow time which gives the MS more time to collect data or allows nodes to operate with higher sampling rates.

4.4. Traversing Algorithms for Mobile Sinks. Based on the *DH* determined by the TDD algorithm and the buffer overflow time derived from the *Allotment Mechanism*, the MS needs to schedule a good traverse path to visit every *DH* in the WSN with the minimum number of missed deadlines. To achieve this goal, we consider a heuristic algorithm and an approximation algorithm, namely, DMWSF and DTSP. Details of these algorithms will be discussed in the following subsections.

4.4.1. Dominating-Based Minimum Weighted Sum First (DMWSF) Algorithm. In the literature [32], the authors use the MWSF algorithm to determine a traverse path for the MS to collect data from all sensor nodes. To integrate with TDD and the *Allotment Mechanism*, we modify MWSF as the DMWSF algorithm, in which the MS will visit *DH*s only with D_o derived from the *Allotment Mechanism*. With fewer nodes to visit, the MS computation load required to find the next visiting *DH* is alleviated. In addition, the increased buffer overflow time D_o results in fewer missed deadlines.

Two factors are considered in the DMWSF algorithm to find the next *DH* to visit. One is the buffer overflow time of

(1) **while** *true* **do**
(2) **Calculate** the weight value for the current *DH* and every other *DH* by (9).
(3) **Choose** the *DH* which has the smallest weight value.
(4) **Move** to the selected *DH* and collect all data from the *DH* and its *DMs*.
(5) **end while**

Algorithm 2: DMWSF algorithm.

(1) **select** a node $r \in V$ to be a root node
(2) **construct** a minimum spanning tree T for G from root r using MST-PRIM(G, w, r)
(3) **let** L be the sequence of vertices in the preorder tree visit of T
(4) **return** the traverse path H that visits the vertices in the order L

Algorithm 3: Dominating-based TSP(G, w).

each *DH*, and the other is the distance between the MS and each *DH*. The MS calculates the weight value for each *DH* based on the following:

$$\text{weight}[i][j] = \alpha \times (D_o[j] - D_o[i]) + (1 - \alpha) \times \text{cost}[i][j]. \quad (9)$$

Here node i is the *DH* that the MS is currently visiting. Weight$[i][j]$ represents the weight value from current node i to the next *DH*, node j. Because sensor nodes have their clocks synchronized, the difference between the buffer overflow time of nodes i and j may represent the imminence of node j's *DC*. A negative value of the difference means the missed deadline of j's *DC*, and it will show the stress when the difference is tiny. On the other hand, the cost$[i][j]$, defined in Section 3, is the distance between node i and j. α has a value between 0 and 1 and is used to adjust the weight between the distance factor and the buffer overflow time factor. If α is bigger than 0.5, the MS decides the next visiting *DH* mainly based on the buffer overflow deadline; otherwise, the MS considers the next *DH* having shorter distance. Note that, when α is equal to 1, DMWSF is equivalent to the Early Deadline First algorithm.

After calculating the weight value for every *DH* by (9), the MS will move to the *DH* with the smallest weight value and collect the sensing data from the *DH* and its *DMs*. When finished, the whole process will start again (refer to Algorithm 2).

Figure 3 illustrates an example of the DMWSF algorithm. Figure 3(a) shows all the *DCs* as determined by the TDD algorithm, and the MS will use *DH*s as rendezvous points. The MS is located close to the position of node A. After calculating weight$[A][B]$ and weight$[A][C]$, the MS will select B as the next visiting node if weight$[A][B]$ < weight$[A][C]$, in Figure 3(b).

The DMWSF algorithm is designed to be integrated with the TDD algorithm, thus, alleviating a problem in the MWSF algorithm in which the MS needs to visit all the sensor nodes without requiring much additional communication. With the *Allotment Mechanism*, it may further reduce the buffer overflow problem by having nodes sharing their memory to buffer data, thus, reducing the number of missed deadlines. The simulation results are presented and discussed in detail in Section 5.

4.4.2. Dominating-Based Traveling Salesman Approximation (DTSP) Algorithm. An intuitive approach to traverse all *DH*s while minimizing the number of missed deadlines is to traverse them in the minimum amount of time. Thus, with a constant moving speed, we want the MS to traverse all *DH*s via the shortest distance. The second traversing algorithm uses a well-known approximation algorithm for TSP [38]. To allow the integration of the TDD and *Allotment Mechanism*, the approximation algorithm is modified and denoted as DTSP, as illustrated in Algorithm 3. Here the traversing problem is modeled as a complete undirected graph $G(V, E)$, and the V is the set of *DH*s and the weight $w[i][j]$ of each edge in E is the derived from (9) for *DH* i and j. Algorithm 3 is widely recognized to be a 2-approximation algorithm since the problems considered are defined in the Euclidean space in which the triangle inequality is satisfied. Note that line 2 of Algorithm 3, *MST-PRIM*, is used to construct a minimum spanning tree for a given graph G. It starts with the root vertex r and chooses the minimum weight edge by w [39].

The MS will continue to visit all *DH*s along the path H. The simulation results are presented and discussed in Section 5.

5. Simulation Environment and Results

To verify the effectiveness and performance of the proposed approach, we conducted various simulations with the GloMoSim [22] network simulator. GloMoSim is a scalable network simulation tool for wired and wireless networks which easily allows the addition of new protocols or the modification of supporting protocols. The algorithms to be compared include Random_Waypoint [16, 31], BISECTOR [17], and MWSF [32]. To serve as a baseline comparison, we also include the data-gathering scheme via static sink, denoted as STATIC.

5.1. Simulation Environment. The objective of our simulations is to compare the number of missed data overflow deadlines and the network lifetime. The network considered is a heterogeneous sensor network with various data-sampling rates. The parameters for the simulation environment are as follows:

(1) ROI: a 500 m $\times$ 500 m square,
(2) Node deployment: 50 $\sim$ 300 sensor nodes uniformly and randomly deployed over the ROI.
(3) data sampling rate: each sensor node is randomly assigned a sampling rate between 0 and 25 (bytes per

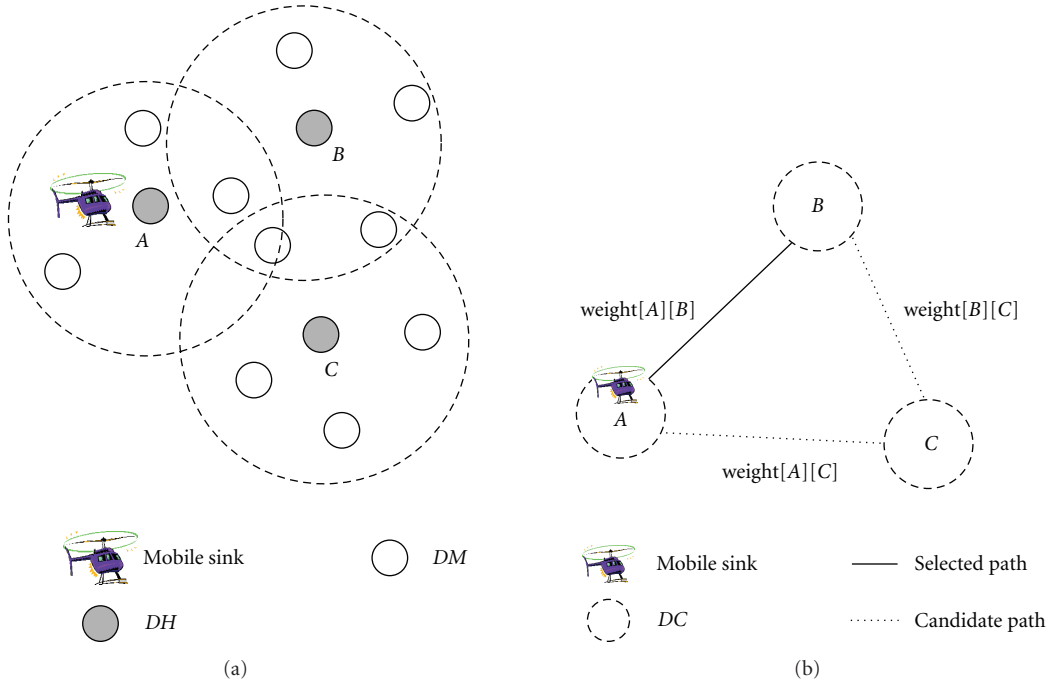


Figure 3: An example of the DMWSF Algorithm. (a) All sensor nodes are partitioned into *DCs*. (b) The MS selects *B* as the next visiting node since weight$[A][B]$ < weight$[A][C]$.

unit of time) and a memory size between 5000 and 10000 bytes,

(4) energy and communication: the initial energy of each node is 8 Joules and the transmission range is 100 m,

(5) mobile sink: the MS has the same transmission range as the sensor nodes, and its velocity is 5 m per unit of time. Memory size and energy are unlimited for the MS.

The medium access control (MAC) protocol applied in our simulations is CSMA. The transmission rate for each node is 19.2 Kb per unit of time. The simulation duration is 10000 units of time.

5.2. Energy Consumption Model. Since wireless communication plays a major role in sensor node energy consumption, we only consider the energy consumed for communication and use the energy consumption model adopted in [23], in which the energy needed to transmit a l-bits packet over a distance d is.

$$E_{tx}(d) = E_{Tx\_elec} \times l + E_{amp} \times l \times d^2, \tag{10}$$

and the energy needed to receive a l-bits packet is,

$$E_{rx} = E_{Rx\_elec} \times l. \tag{11}$$

The values of the parameters are listed in Table 2.

5.3. The α Value . The α value of (9) controls the priority of the data overflow time and the inter-distance of the *DH*s. If $\alpha > 0.5$, the buffer overflow time will have a higher priority than the interdistance, and vice versa. To determine which factor has a greater impact on performance, we first conducted simulations to evaluate how α affects the maximum tolerable sampling rate, that is, the maximum sampling rate without any missed deadlines, and the traverse distance.

Table 2: Energy consumption for wireless communication.

Operation	Energy consumption
Transmit electronics (E_{Tx_elec}) Receive electronics (E_{Rx_elec})	50 nJ/bit
Transmit amplifier (E_{amp})	100 pJ/bit/m^2

100 sensor nodes were uniformly and randomly deployed over the ROI. Other parameters of each sensor node are described in Section 5.1. Figure 4 illustrates the simulation results for various α values from 0.05 to 0.9. In Figure 4(a), the y-axis is the maximum tolerable sampling rate (bytes per unit of time). This shows that the WSN's maximum tolerable sampling rate is the lowest when $\alpha = 0.05$ and 0.9 and the highest when α is between 0.4 and 0.5. Thus, to effectively avoid the buffer overflow problem, we should simultaneously consider both factors (i.e., buffer overflow time and inter-distance), as ignoring one of these factors will lead to the most severe buffer overflow problem.

In Figure 4(b), the y-axis is the traverse distance of the MS. This shows that the traverse distance is shorter when α is smaller. This is not surprising since the inter-distance has higher priority over buffer overflow time when α is small. Based on these results, we set $\alpha = 0.5$ in the

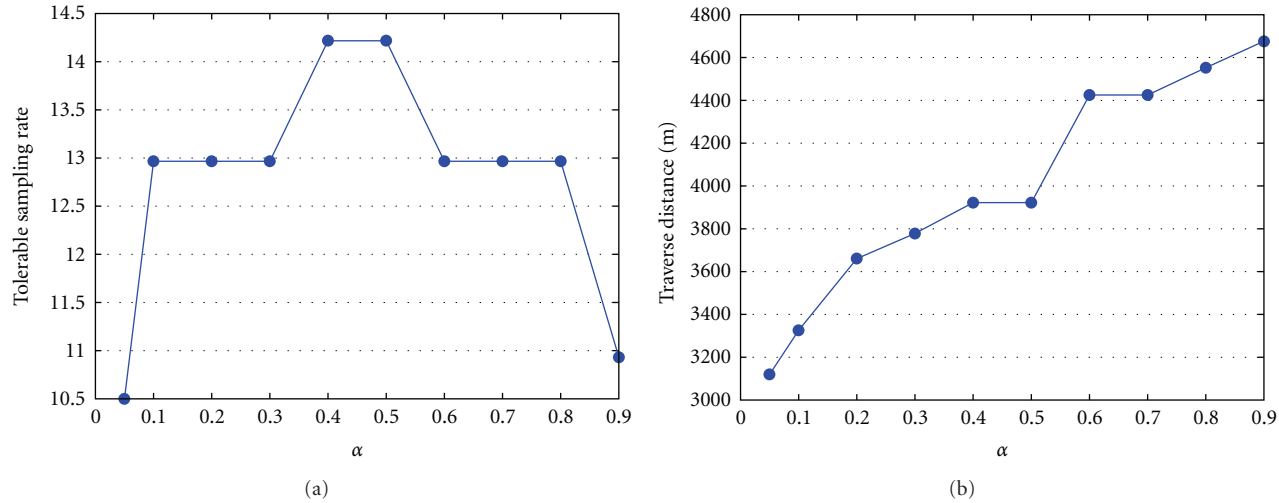


FIGURE 4: (a) The WSN's maximum tolerable sampling rate (bytes per unit of time) is the lowest when $\alpha = 0.05$ and 0.9 and the highest when α is between 0.4 and 0.5. (b) A smaller α results in shorter traverse distances for the MS.

subsequent simulations so the tolerable sampling rate is maximized without much significantly affecting the MS's traverse distance.

5.4. Maximum Tolerable Sampling Rate and Traverse Distance. Figure 5(a) compares the maximum tolerable sampling rate against the number of nodes for various data-gathering algorithms, namely, BISECTOR, MWSF, DMWSF with various k values and DTSP with various k values. Here the number of nodes is between 50 and 300, and k is the number of *DM*s that share their memory space for the *Allotment Mechanism*. When $k = 0$, the *Allotment Mechanism* is not applied.

As seen in Figure 5(a), the maximum tolerable sampling rate decreases (i.e., results in more severe buffer overflow problems) as the number of nodes increases. This is not surprising since there may be more *DH*s to visit as the number of nodes increases and the sampling rate needs to be reduced to meet all the deadlines. Besides, the larger value of k (i.e., $k = 2$) will lead to a higher maximum tolerable sampling rate, which indicates that the *Allotment Mechanism* can actually alleviate the buffer overflow problem. However, the performance is improved more significantly when the k value increases from zero to one than when it increases from one to two, which suggests that $k = 1$ may be an appropriate value for the *Allotment Mechanism* since *DH*s have more energy consumption overhead for larger k.

In general, DMWSF and DTSP provide similar performance. On the other hand, the BISECTOR algorithm gathers data by designating turning points. Sensor nodes need to transmit data to one of these turning points via multihop forwarding, which leads to the hot spot problem. This can be alleviated by using lots of turning points (i.e., more than the *DH*s used in DMWSF and DTSP), to meet all buffer overflow deadlines. Thus, because BISECTOR has more points to traverse, its maximum tolerable sampling rate is lower than that of DMWSF or DTSP. Finally, MWSF has the lowest maximum tolerable sampling rate since it needs to traverse all sensor nodes. Note that for each simulation nodes are deployed over the same ROI. That is, in MWSF, the MS simply travels the entire ROI to visit every node. Thus, the maximum tolerable sampling rate is independent of the number of nodes but dependent on the size of the ROI.

Figure 5(b) compares the MS's traverse distance against the number of nodes for various data-gathering algorithms. Intuitively, if an algorithm has a shorter traverse distance for the MS, it may have a more tolerable higher sampling rate without any buffer overflow. In MWSF, the MS needs to visit all sensor nodes and, thus, has the longest traverse distance. Note that, in [32], sensor nodes are deployed within concentric circles in which the nodes in the innermost region have the lowest sampling rates and the outer regions can have higher sampling rates. Such a deployment and sampling rate designation allows for a short traverse path. However, in general deployment, MWSF does not provide short traverse distances for the MS.

Figure 5(b) also shows that the traverse distances of BISECTOR, DMWSF, and DTSP increase slightly with the number of nodes. This is because nodes are deployed on the same ROI, and increasing the number of nodes over the same region increases the density but may only slightly increase the number of *DH*s or turning points. Thus, the traverse distances do not increase significantly. DMWSF and DTSP outperform BISECTOR and MWSF in this metric as well.

Therefore, the dominating-based algorithms, DMWSF and DTSP together with TDD and the *Allotment Mechanism*, can alleviate the buffer overflow problem for randomly deployed heterogeneous sensor networks.

5.5. Lifetime and Energy Consumption. In this section, we compare energy consumption performance for the various algorithms. The ROI contains 50–200 sensor nodes deployed uniformly and randomly. Each sensor node has a data-sampling rate of either 2 or 4 bytes per unit of time. In addition to the data-gathering algorithms mentioned in the

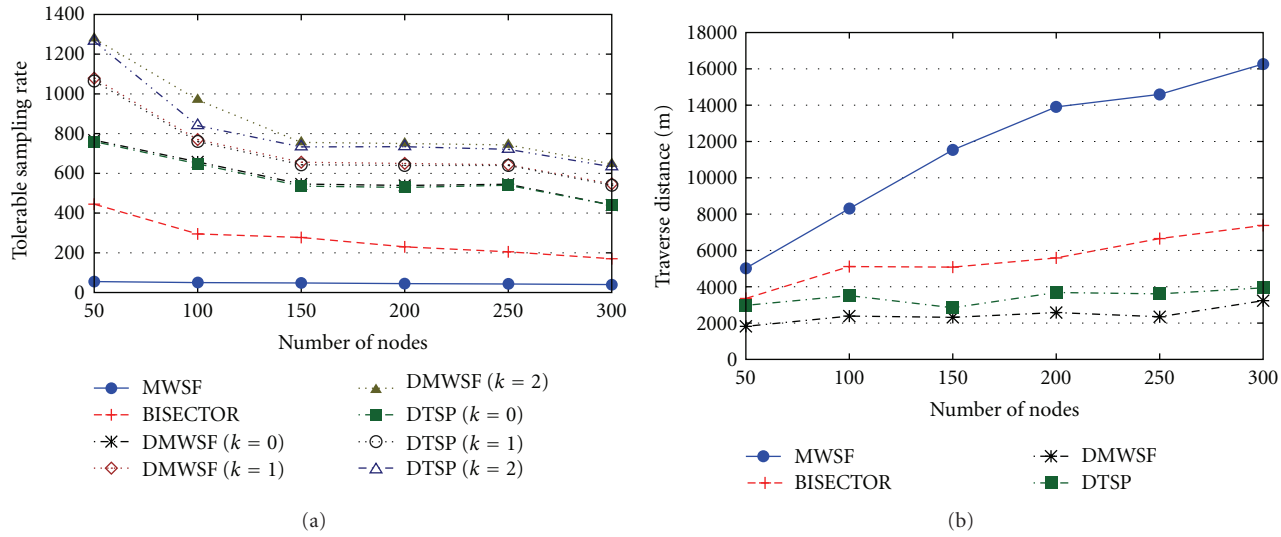


FIGURE 5: (a) Maximum tolerable sampling rate (bytes per unit of time) for each data-gathering approach. (b) MS traverse distance for each data-gathering approach.

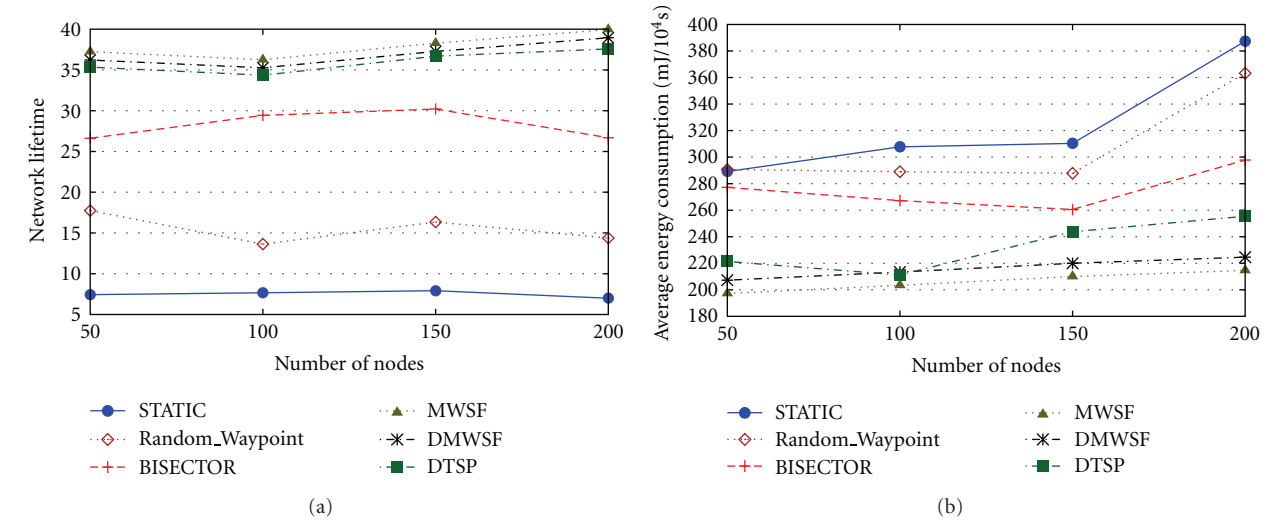
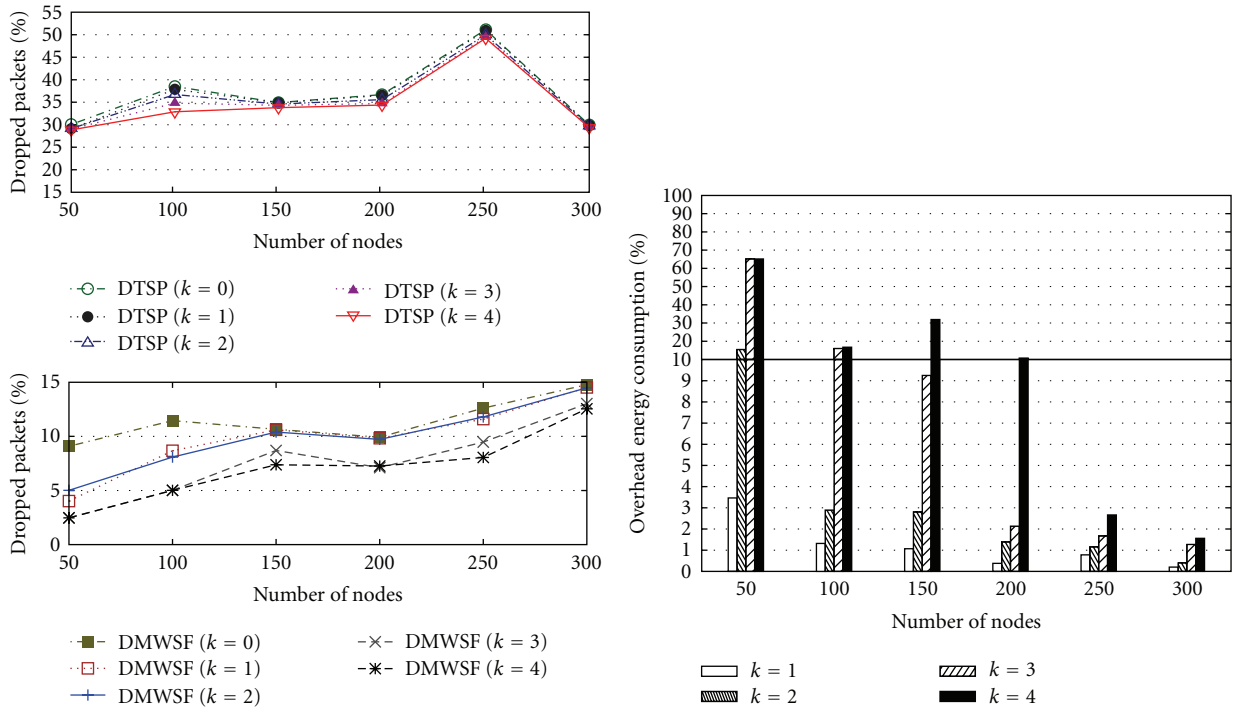


FIGURE 6: (a) Network lifetime (in units of time) for each data-gathering approach. (b) Average energy consumption for each data-gathering approach.

previous subsection, we also include Random_Waypoint and STATIC. In Random_Waypoint, the MS randomly moves straight to a randomly selected destination in the ROI where it collects data from nearby sensor nodes. In STATIC, the data are collected by a static sink via multihop forwarding.

Figure 6(a) compares the network lifetime (in units of time) against the number of nodes for various data-gathering algorithms. Here, the lifetime is defined as the time it takes for a sensor node to exhaust its energy. As Figure 6(a) indicates, STATIC has the shortest network lifetime because of the hot spot problem. Note that Random_Waypoint uses the MS moving straight to a randomly selected destination, but some sensor nodes near the MS also need to relay packets for the other nodes; thus, the hot spot problem still occurs, though with less severity than in STATIC. Thus, the network lifetime of Random_Waypoint is better than that of STATIC, but worse than others. In a heterogeneous WSN, BISECTOR works the same way as in a WSN in which each node has the same sampling rate (i.e., the MS moves from one turning point to another to collect data from sensor nodes via multihop forwarding). Though the hot spot problem still occurs, the presence of multiple turning points in BISECTOR results in better network lifetime than in Random_Waypoint and STATIC.

Note that MWSF has the longest network lifetime among all data-gathering approaches. This is not surprising since the MS will move to each node to collect data. Thus, there is no need for data forwarding for MWSF. However, as indicated in the previous subsection, MWSF has the lowest maximum tolerable sampling rate, which means that MWSF



(a) (b)

FIGURE 7: (a) Percentage of dropped packets, defined as dropped packets/total packets ×100%, for different k values. (b) Percentage of overhead energy consumption, defined as energy consumption of *Allotment Mechanism*/total energy consumption ×100%, for different k values.

has the worst buffer overflow problem. On the other hand, in DMWSF and DTSP, the MS only needs to traverse *DH*s and collect the data from *DM*s via one-hop forwarding. Thus, the energy consumption of DMWSF and DTSP should be close to that of MWSF; that is, DMWSF and DTSP have slightly shorter network lifetimes than does MWSF, as seen in Figure 6(b). However, as discussed in the previous subsection, both DMWSF and DTSP have much better maximum tolerable sampling rates than does MWSF, which makes them promising schemes for data gathering. Figure 6(b) compares the average energy consumption against the number of nodes for the various data-gathering algorithms. It is not surprising that STATIC has the worst energy consumption since it requires so much multihop forwarding. The energy consumption of Random_Waypoint is also high for the same reason. As in Figure 6(b) above, MWSF has the best energy efficiency, closely followed by DMWSF and DTSP. Again, considering the buffer overflow problem, DMWSF and DTSP are promising schemes for data gathering.

5.6. Performance of the Allotment Mechanism with Different k Values. We also conducted simulations to illustrate the impact of k values on the *Allotment Mechanism*. Here, each sensor node is randomly assigned a sampling rate between 0 and 25 bytes per unit of time, and the duration is 10000 units of time. We compared the performance of DMWSF and DTSP with k = 0–4. Figure 7(a) shows the percentage of dropped packets due to buffer overflow against the total number of nodes. As indicated, the percentage of dropped packets decreases as the k value increases. However, as illustrated in Figure 7(b), the percentage of overhead energy consumption indicates that the buffer overflow problem is alleviated at the cost of increased energy consumption. In our experiments, the *Allotment Mechanism* with $k \geq 3$ can no longer effectively alleviate the buffer overflow problem but only consumes more energy.

6. Conclusions and Future Work

Typical WSNs use static sinks to collect data from all sensor nodes via multihop forwarding, which can easily result in the hot spot problem since nodes close to the sink tend to consume more energy in relaying data from other nodes. This can exhaust the close nodes, leaving the sinks isolated from the rest of network and the remaining nodes underutilized.

An MS can prevent the hot spot problem, but it takes time to move around the ROI to collect data. A poorly designed traverse path may result in the buffer overflow problem since the MS cannot arrive at nodes in time to collect the data buffered in their memory, necessitating the dropping of some information.

Our proposed approach addresses the hot spot problem using an MS to collect data, but the MS only traverses the rendezvous points, achieved by TDD, where every node is within a one-hop communication range of a rendezvous

point (i.e., the dominating set). Reducing the number of points to traverse reduces the time needed to traverse them, thus, alleviating the buffer overflow problem. The traverse path is determined by DMWSF or DTSP, in which the traverse cost consists two factors: the buffer overflow time and interdistance. The weighting between these two factors is controlled by the α value. Furthermore, we proposed the *Allotment Mechanism* that allows the nodes with higher sampling rates to temporarily buffer their data in the memory of their one-hop neighbors with lower sampling rates, thus, extending the buffer overflow deadline which further alleviates the buffer overflow problem.

The effectiveness of proposed approach was verified via the GloMoSim network simulator. Simulation results show that our approach incurs fewer buffer overflows than other data-gathering schemes such as BISECTOR and MWSF. Moreover, the simulation results of α value test suggest that $\alpha = 0.4$–0.5 has the least buffer overflow problem, which means that both buffer overflow time and interdistance need to be considered when planning a traverse path for an MS. In addition, the buffer overflow problem can be alleviated with a larger k at the cost of increased energy consumption. However, our simulation results show that the *Allotment Mechanism* with $k \geq 3$ can no longer effectively alleviate the buffer overflow problem but only consumes more energy.

Finally, in future work we plan to design a more efficient *Allotment Mechanism* for large-scale wide sensor network environments, for example, using a topology control supported in the IEEE 802.15.4 standard [40]. We will also consider the possibility of using adaptive k values. For example, information loss may result if a *DH* is too far from the initial position of the MS. We will study whether this problem can be alleviated by using k correlated with the distance from the *DH* to the initial position of the MS. In addition, we will investigate more recent and efficient simulation platforms, such as the OMNeT++ simulator [41] which gives a more realistic behavior in WSNs. With more realistic propagation models and different MAC protocols, we may verify the effectiveness of our proposed approach under unreliable communications conditions.

Acknowledgment

This research was supported by National Science Council (NSC), Taiwan, ROC, under Grant NSC 99-2221-E-194-021. The authors gratefully acknowledge this support.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] A. Beaufour, M. Leopold, and P. Bonnet, "Smart-tag based data dissemination," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 68–77, ACM, September 2002.

[3] J. R. Polastre, *Design and implementation of wireless sensor networks for habitat monitoring*, M.S. thesis, University of California at Berkeley, 2003.

[4] A. Chehri, P. Fortier, and P.-M. Tardif, "Security monitoring using wireless sensor networks," in *Proceedings of the 5th Annual Conference on Communication Networks and Services Research*, pp. 13–17, IEEE Computer Society, Washington, DC, USA, 2007.

[5] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, vol. 37, pp. 96–107, October 2002.

[6] W. Du, L. Fang, and N. Peng, "LAD: localization anomaly detection for wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 66, no. 7, pp. 874–886, 2006.

[7] H. W. Tsai, C. P. Chu, and T. S. Chen, "Mobile object tracking in wireless sensor networks," *Computer Communications*, vol. 30, no. 8, pp. 1811–1825, 2007.

[8] B. Sun, L. Osborne, Y. Xiao, and S. Guizani, "Intrusion detection techniques in mobile ad hoc and wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 5, pp. 56–63, 2007.

[9] T. Miyazaki, R. Kawano, Y. Endo, and D. Shitara, "A sensor network for surveillance of disaster-hit region," in *Proceedings of the 4th International Symposium on Wireless and Pervasive Computing*, pp. 1–6, February 2009.

[10] T. Park, D. Kim, S. Jang, S. E. Yoo, and Y. Lee, "Energy efficient and seamless data collection with mobile sinks in massive sensor networks," in *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium*, pp. 1–8, May 2009.

[11] Y. S. Chen, S. Y. Ann, and Y. W. Lin, "VE-mobicast: a variant-egg-based mobicast routing protocol for sensornets," *Wireless Networks*, vol. 14, no. 2, pp. 199–218, 2008.

[12] M. Halkidi, V. Kalogeraki, D. Gunopulos, D. Papadopoulos, D. Zeinalipour-Yazti, and M. Vlachos, "Efficient online state tracking using sensor networks," in *Proceedings of the 7th International Conference on Mobile Data Management*, IEEE Computer Society, Washington, DC, USA, 2006.

[13] C. Weng, M. Li, and X. Lu, "Data aggregation with multiple spanning trees in wireless sensor networks," in *Proceedings of the International Conference on Embedded Software and Systems*, pp. 355–362, July 2008.

[14] T. L. Sheu and W. C. Liu, "An adaptive data collection scheme for mobile sinks in a grid-based wireless sensor network," in *Proceedings of the 3rd International Conference on Communications and Networking in China*, pp. 382–386, August 2008.

[15] R. Yu, X. Wang, and S. Das, "Efficient data gathering using mobile elements in partially connected sensor networks," in *Proceedings of the Chinese Control and Decision Conference*, pp. 5337–5342, July 2008.

[16] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling a three-tier architecture for sparse sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 30–41, 2003.

[17] M. Ma and Y. Yang, "SenCar: an energy-efficient data gathering mechanism for large-scale multihop sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 10, pp. 1476–1488, 2007.

[18] G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing2008*, pp. 231–239, May 2008.

[19] D. Mandala, X. Du, F. Dai, and C. You, "Load balance and energy efficient data gathering in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 8, no. 5, pp. 645–659, 2008.

[20] D. Cokuslu, K. Erciyes, and O. Dagdeviren, "A dominating set based clustering algorithm for mobile ad hoc networks," in *Proceedings of International Conference on Computational Science*, pp. 571–578, 2006.

[21] J. Wu and H. Li, "A dominating-set-based routing scheme in ad hoc wireless networks," *Telecommunication Systems*, vol. 3, pp. 63–84, 1999.

[22] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "GloMoSim: a scalable network simulation environment," Tech. Rep. 990027, UCLA Computer Science Department, 1999.

[23] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, IEEE Computer Society, Washington, DC, USA, 2000.

[24] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach," in *Proceedings of the 23rd Conference of the IEEE Communications Society*, pp. 629–640, March 2004.

[25] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C.-G. Lee, "Partitioning based mobile element scheduling in wireless sensor networks," in *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pp. 386–395, 2005.

[26] R. Wohlers, N. Trigoni, R. Zhang, and S. A. Ellwood, "TwinRoute: energy-efficient data collection in fixed sensor networks with mobile sinks," in *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, pp. 192–201, 2009.

[27] G. S. Chhabra and D. Sharma, "Cluster-tree based data gathering in wireless sensor network," *International Journal of Soft Computing and Engineering*, vol. 1, pp. 27–32, 2011.

[28] K. D. Samuel, S. M. Krishnan, K. Y. Reddy, and K. Suganthi, "Improving energy efficiency in wireless sensor network using mobile sink," in *Advances in Networks and Communications*, N. Meghanathan, B. K. Kaushik, and D. Nagamalai, Eds., vol. 132 of *Communications in Computer and Information Science*, pp. 63–69, Springer, Berlin, Germany, 2011.

[29] Z. Chen, S. Liu, and J. Huang, "Multi-tier grid routing to mobile sink in large scale wireless sensor networks," *Journal of Networks*, vol. 6, pp. 765–773, 2011.

[30] L. He, Z. Chen, and J.-D. Xu, "Optimizing data collection path in sensor networks with mobile elements," *International Journal of Automation and Computing*, vol. 8, pp. 69–77, 2011.

[31] S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy, "Exploiting mobility for energy efficient data collection in wireless sensor networks," *Mobile Networks and Applications*, vol. 11, no. 3, pp. 327–339, 2006.

[32] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Proceedings of the 25th IEEE International Real-Time Systems Symposium, Washington*, pp. 296–305, IEEE Computer Society, Washington, DC, USA, 2004.

[33] J. Rao and S. Biswas, "Data harvesting in sensor networks using mobile sinks," *IEEE Wireless Communications*, vol. 15, no. 6, pp. 63–70, 2008.

[34] E. M. Saad, M. H. Awadalla, and R. R. Darwish, "A data gathering algorithm for a mobile sink in large-scale sensor networks," in *Proceedings of the 4th International Conference on Wireless and Mobile Communications*, pp. 207–213, August 2008.

[35] F. Dai and J. Wu, "Distributed dominant pruning in Ad Hoc networks," in *Proceedings of the 2003 International Conference on Communications*, pp. 353–357, May 2003.

[36] B. Han, H. Fu, L. Lin, and W. Jia, "Efficient construction of connected dominating set in wireless ad hoc networks," in *Proceedings of the 2004 IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 570–572, October 2004.

[37] Y. Bi, J. Niu, L. Sun, W. Huangfu, and Y. Sun, "Moving schemes for mobile sinks in wireless sensor networks," in *Proceedings of the 27th IEEE International Performance Computing and Communications Conference*, pp. 101–108, April 2007.

[38] Concorde TSP Solver. http://www.tsp.gatech.edu/concorde.html.

[39] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Systems Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[40] C. Buratti, M. Martalò, R. Verdone, and G. Ferrari, *Sensor Networks with IEEE 802.15.4 Systems*, Springer, 2011.

[41] OMNeT++ Network Simulation Framework. http://www.omnetpp.org/.