Sink Mobility Strategies for Reliable Data Collection in Wireless Sensor Networks

Yuki Fujita*, Daichi Kominami† and Masayuki Murata*
*Graduate School of Information Science and Technology, Osaka University, Japan
{y-fujita, murata}@ist.osaka-u.ac.jp
†Graduate School of Economics, Osaka University, Japan
d-kominami@econ.osaka-u.ac.jp

***Abstract*—The Internet of Things and machine-to-machine communications will form one of the most important backbones of our life in the near future. Therefore, more reliability is required in many wireless sensor network applications, such as structural health monitoring systems, intruder detection systems, and search and rescue systems. However, without the assumption that all sensor nodes can reach a sink node through multi-hop communication and that the connectivity among all sensor nodes is stable, it is difficult to guarantee the reliability of data collection. In this paper, we focus on controlling the mobility of a mobile sink and propose two types of mobility strategies to collect sensing data certainly from all sensor nodes in an observed area. One strategy is to learn the positions of all isolated networks in the observed area and the other is to collect sensing data using the learned positions. Through computer simulations, we show that the mobile sink with the mobility strategies can collect the sensing data from all sensor nodes.**

Keywords–Wireless sensor networks, reliable data collection, mobile sink, controlled mobility.

I. INTRODUCTION

Supporting assured data collection in wireless sensor networks (WSNs) is one of the significant challenges in frequently changing environments. This is because dynamic changes in the observed area and loss of reachability to sink nodes occur in actual situations, which cannot be dealt with through conventional transport techniques. This promotes network-level reliable mechanisms for data collection. We focus on the mobility control for a data collecting node, usually called a *sink node* in WSNs, to realize reliable data collection.

Wireless sensor networks, which facilitate the collection of environmental information, are expected to apply significantly to various applications, e.g., structural health monitoring of infrastructures, monitoring of temperature and humidity on a farm, tracking of animals, etc. [1], [2]. In many cases, WSNs are composed of many sensor nodes and a few sink nodes, which operate in a distributed manner and are connected to each other. Sensor nodes forward their sensing data to one of the sink nodes through multi-hop wireless communications, which makes it possible to collect various environmental information.

In the near future, many machines will be mutually connected and will be quietly embedded in our life space. Thus, the Internet of Things and machine-to-machine communications will make WSN techniques more and more significant. In applications strongly tied to safety and security, the reliability of data gathering is one of the most important viewpoints. Data collection is realized under the assumption that all sensor nodes are reachable by one of the sink nodes through multi-hop communication. However, this assumption is not always realistic due to the limitation of the communication range of nodes, changes of wireless channel conditions, or failures of sensor nodes. It is inappropriate to allow nodes to directly communicate with a sink node since sensor nodes have a limited battery capacity in most WSNs. Also, it is difficult to deploy sensor nodes over the observed area with paying excess attention that all sensor nodes are always reachable to a sink node.

We focus on a sink node with mobility called a *mobile sink*. A mobile sink can achieve both reduction of power consumption and reachability of every sensor nodes by approaching each sensor node, receiving data, and carrying it to the static base station. Many studies have been conducted about mobile sinks as a solution for power saving, which is one of the challenging problems in WSNs, such as path planning of mobile sinks and efficient data routing algorithms considering the movement of a mobile sink [3]–[5]. Mobile sinks from the viewpoint of reliable data gathering do not have as much active research.

Controlled mobility is a key idea for maintaining network connectivity and achieving data reachability for users, where the mobility of mobile sinks is dynamically controlled from both inside and outside of networks [6], [7]. We previously combined controlled mobility with a potential-based routing mechanism in a wireless sensor network, where periodically transmitted route information messages lead a mobile sink toward the static data collecting node (called the target node) and the mobile sink receives all data from the target node [8]. In this method, a mobile sink can collect data from a network, however, the mobile sink cannot deal with the loss of reachability to the target node.

In this paper, we propose two mobility strategies for a mobile sink for realizing reliable data collection. To this end, we need to manage the following two changes in an observed area caused by failures, energy depletion, or additions of sensor nodes. Here, we define a *sub-network* as a set of all sensor nodes reachable from each other by multi-hop communication.

- Small changes in a sub-network, which do not increase

 ISBN: 978-1-61208-391-9

or decrease the number of sub-networks, but cause changes in route. This occurs mainly due to link failures and node failures.

- Large changes in a sub-network, which increase or decrease the number of networks and also cause changes in route. This occurs for various reasons that cause the disconnection of a sub-network, the jointing of sub-networks, or the deployment of a new sub-network.

Small changes have been well-studied, however, large changes have been little considered in existing studies. Thus, our interests are in how we can collect *all* data in an observed area when both types of changes occur. We use the term *reliable data collection* as 100% collection of data that are generated by all sensor nodes in each sub-network. We aim for reliable data collection by using a mobile sink within an observed area where both the number and the positions of sensor nodes are unknown. In these situations, it is a possible (but not practical) method for a mobile sink to travel all over the observed area since the mobile sink does not know where sub-networks are in the observed area. Of course, it takes much more time for the mobile sink to travel to every nook and cranny as the observed area gets larger. Therefore, the first mobility strategy is conducted at long intervals, where a mobile sink travels all over the observed area to grasp all positions of sub-networks and also impassable locations. The other strategy is to visit all sub-networks and to collect sensing data from data possessing nodes using learned positions from the first strategy.

In principle, it is difficult to catch an unexpected change by methods other than the first strategy, and it requires a lot of time. Therefore, it is taken repeatedly over a long period. For the second strategy, we use a clustering technique and all data in a sub-network are gathered in one or more cluster heads. Then, a mobile sink just has to visit such cluster heads to collect data. Note that we assume that cluster heads change their role back to a non-cluster head periodically for managing small changes in a sub-network and for achieving load balancing. Thus, a mobile sink does not always know the position of a cluster head. The controlled mobility mechanism proposed in [8] helps a mobile sink approach a cluster head in each sub-network.

The remainder of this paper is organized as follows. In Section II, we present the mobility control strategy for memorizing locations of networks. In Section III, we show the mobility strategy for collecting sensing data in a network. Section IV presents simulation results, and finally, we conclude our paper in Section V.

II. Mobility Strategy for Memorizing Sub-Network Locations

A mobile sink has to periodically check the entire picture of the observed area, such as the positions of sub-networks and forbidding places, to determine the path for visiting all sub-networks. In order to grasp this information, it moves over the entire observed area while identifying and memorizing all the different sub-networks.

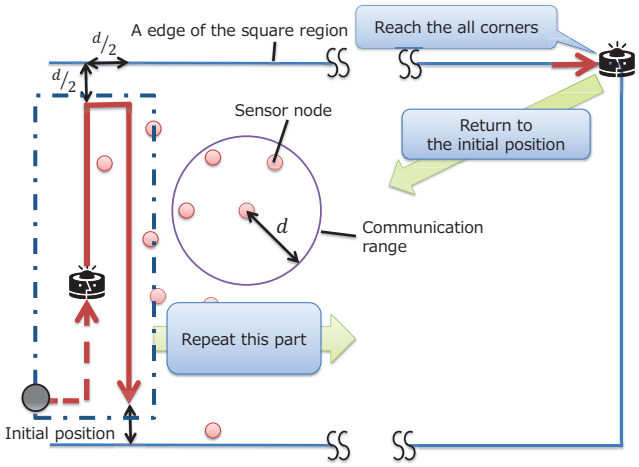

Figure 1. Mobility strategy for detecting all sensor nodes without any oversight

In our proposal, a mobile sink moves so that it does not overlook even one sensor node placed in the observed area. To begin with, we assume that only a mobile sink has a positioning device like a global positioning system. A mobile sink commences to move from a given initial position (e.g., a base station with charging capability for a mobile sink battery), which is one of the corners of the pre-defined square region including the whole observed area as illustrated in Fig. 1. Then, the mobile sink goes straight on toward one nearby corner until it reaches $d/2$ length short of the corner, where d is the wireless communication range of the mobile sink and sensor nodes. Then, it takes a turn toward the other nearby corner, moves ahead $d/2$, and again rotates in the same angle and moves ahead. The mobile sink repeats the same process until reaching all corners, then it returns to the initial position.

In this strategy, a mobile sink intercepts a message **PInfoMsg**, which all sensor nodes transmit and exchange with each other for updating route information (described in Section III-B in detail). Then, the mobile sink acquires a special identifier (ID) contained in that **PInfoMsg**, which is used to identify sub-networks.

A mobile sink memorizes or updates a position of sub-network according to Algorithm 1, where some terms are listed in Table I. A mobile sink has a table **NetTable** for storing sub-network positions, which is updated every time it receives a **PInfoMsg**. A **NetTable**'s entry is the tuple ($netID$, $position$, RSSI) where $netID$ is an identifier of a sub-network, $position$ is the position where the mobile sink received the **PInfoMsg**, and RSSI is the received signal strength indication of the **PInfoMsg**. A new entry is always registered to the table if there is no entry whose $netID$ equals one in the **PInfoMsg**, and an existing entry is updated if the RSSI of the received **PInfoMsg** is greater than an existing one that has the same $netID$ of the **PInfoMsg**. This is for ensuring that the mobile

 ISBN: 978-1-61208-391-9

Algorithm 1 Memorizing the positions of networks associating with $netID$ by the mobile sink

```
// The mobile sink moves in every corner of the observed area.
repeat
  if intercepts PInforMsg(i, netID_i, pList_i, vList_i) then
    if NetTable has no entry with netID_i then
      register NetTable(netID_i, pos, PInforMsg.rssi)
    else
      if PInfoMsg.rssi > entry.rssi then
        update NetTable(netID_i, pos, PInforMsg.rssi)
      end if
    end if
    if State_i is CLUSTER(i, 0) then
      sends SensingDataRequest to S_i
    end if
  end if
until reaches the end of the observed area
```

TABLE I. NOTATIONS IN OUR PROPOSED METHODS

Notation	Description
N_s	The number of sensor nodes which is initially deployed
N_b	The number of sensor nodes which will get failed
N_a	The number of sensor nodes which will be added
S_i	The sensor node whose ID is i
$ND(S_i)$	The number of neighbor nodes of S_i
$State_i$	The state of S_i which indicates whether S_i belongs to a cluster or not. $State_i$ is either **UNCLUSTER** or **CLUSTER**(i, n)
t_{S_i}	The current time of node S_i
T_{lim}	The time limit to search for its neighbor nodes
T_{flood}	The interval that cluster heads broadcast **PInfoMsg**
T_h	The period that **NTable** keeps a non-updated $entry$
$netID_i$	The ID of the network that S_i belongs to
pID	The ID of the potential field
myP_i	The potential values that S_i has
$pList_i$	The set of pID that S_i has
$vList_i$	The set of myP_i that S_i has

sink can obtain more accurate positions of sub-networks. Note that, when the mobile sink can contact a cluster head in this mobility strategy, it demands sensing data from the cluster head by transmitting a message **SensingDataRequest**.

III. MOBILITY STRATEGY FOR VISITING ALL CLUSTER HEADS

In the previous strategy, a mobile sink memorizes the position of all sub-networks. In order to realize reliable data collection, the mobile sink has to visit each sub-network, collect all sensing data in the sub-network, and then move ahead to another sub-network. One way for collecting all sensing data in a sub-network is to visit sensor nodes one by one by memorizing the position of each node in the previous strategy, which spends memory costs and time costs. In our proposal, all sub-networks within the observed area have one or more special nodes that gather sensing data from all sensor nodes in an individual sub-network, and a mobile sink moves to and sojourns with them to bring the sensing data to the initial position. We elect this special node by using a cluster head election algorithm proposed in [9]. When a mobile sink enters a sub-network, the mobile sink intercepts and interprets route information messages exchanged with sensor nodes and moves in the same direction as data flows, and consequently, it can visit all cluster heads in the sub-network. After that, the mobile sink moves ahead in the direction of another sub-network learned at the previous strategy.

A. Cluster heads election

We elect one or more cluster heads for each sub-network by using a part of the DEECIC algorithm [9] with minor modification. The cluster head election algorithm in DEECIC is described in Algorithm 2 with some terms which are tabulated in Table I. First, sensor node S_i broadcasts an **UpdatePacket** to notify its neighbors of its presence at randomly chosen time t $(0 < t < T_{lim})$. Then, S_i broadcasts a **DegreePacket** including $ND(S_i)$, which is the number of received **UpdatePackets** until T_{lim} expires, to inform its node of its degree at t $(T_{lim} \leq t < T_{lim} + T_{S_i})$. T_{S_i} is given from $T_{S_i} = \alpha e^{1/ND(S_i)}$ where α is a constant so that it ensures $0 < T_{S_i} \ll T_{lim}$. Sensor node S_i waits for receipt of a **StatePacket** including the state of a neighbor node, which notifies whether the neighbor belongs to any cluster or not. Here, **CLUSTER**(i, n) presents that the node S_i can reach a certain cluster head within n hops and CLUSTER$(i, 0)$ means that node S_i is a cluster head. Upon receiving a **StatePacket** with **CLUSTER**(s, n), S_i sets $State_i$ to **CLUSTER**$(i, n + 1)$ if $State_i$ was **UNCLUSTERD** or **CLUSTER**(i, m) where m is larger than $n + 1$. S_i broadcasts a **StatePacket** if $n + 1 \leq max\_n$ thereafter, which limits the coverage of each cluster.

When disconnection of a sub-network occurs, there may be no cluster head in a sub-network. Therefore, this cluster election is done periodically. Thus, our proposed method can respond to environmental changes inside a sub-network.

B. Construction and update of potential fields

We use a potential-based routing protocol [10] for data collection in each sub-network. The potential-based routing is known as a resilient routing protocol for environmental variations. Every node updates its own potential, which is a scalar value calculated only with local information—own potential, neighbors' potential, and node degrees. It is worth noting that potential messages exchanged between sensor nodes for data routing is also utilized for the guidance of a mobile sink toward elected cluster heads.

For the mobility strategy to visit all cluster heads, every cluster head constructs one potential field that is the shape of a concave curve whose bottom corresponds to the position of the cluster head. Thus, multiple potential fields can be constructed in each sub-network. Each potential field has a unique identifier pID that corresponds to an identifier of the

 ISBN: 978-1-61208-391-9

Algorithm 2 Selection of cluster heads in a network

1: // all nodes perform following;
2: $State_i \Leftarrow$ **UNCLUSTER**
3: $t \Leftarrow$ random value between 0 and T_{lim}
4: broadcast **UpdatePacket** at t_{S_i}
5: **repeat**
6: **if** receives a **UpdatePacket then**
7: $ND(S_i) \Leftarrow ND(S_i) + 1$
8: **end if**
9: **until** $t_{S_i} \geq T_{lim}$
10: broadcast **DegreePacket** at t, $(T_{lim} \leq t < T_{lim} + T_{S_i})$
11: **repeat**
12: **if** receive **StatePacket** with **CLUSTER**(s, n) node **then**
13: **if** $State_i$ is **UNCLUSTER then**
14: $State_i \Leftarrow$ **CLUSTER**$(s, n+1)$
15: **else if** $State_i$ is **CLUSTER**(i, m) and $m > n+1$ **then**
16: $State_i \Leftarrow$ **CLUSTER**$(s, n+1)$
17: **end if**
18: **if** n+1 $\leq max\_n$ **then**
19: broadcast **StatePacket**
20: **end if**
21: **end if**
22: **until** $t_{S_i} \geq T_{lim} + T_{S_i}$
23: **if** $ND(S_i)$ is larger than all neighbor nodes and $State_i$is **UNCLUSTER then**
24: $State_i \Leftarrow$ **CLUSTER**$(i, 0)$
25: broadcast **StatePacket**
26: **end if**

cluster head that is a owner of the potential field. The potential-field construction process is given in Algorithm 3 with some terms tabulated in Table I.

First, cluster head S_i initializes $netID_i$ and myP_i, and then broadcasts a **PInfoMsg** throughout the sub-network (lines 1–10). A **PInfoMsg** includes the sender's ID (S_i), sub-network ID, multiple potential-field IDs (pID), and potential values of correspondent potential fields as illustrated in Fig. 2. When receiving a **PInfoMsg**, a sensor node updates the **NTable**, myP_i, and $netID_i$ and broadcasts a new **PInfoMsg** (lines 11–44). Here, **NTable** is a table to store information about the potential of neighbor nodes, and its entry is composed of five attributes $(src, networkID, pID, pValue, time)$, which are an ID of the source node (cluster head), a sub-network ID, a potential-field ID, a potential value in the correspondent potential field, and the time this entry was registered or updated, respectively. S_i registers or updates an entry of their **NTable** when receiving a **PInfoMsg** and S_i removes an old entry that is not updated for a period T_h. After that, S_i calculates its own sub-network ID ($netID_i$) and potentials (myP_i). $netID_i$ is set to the lowest value among pID registered in its **NTable** and myP_i is set to an average potential value of its neighbors as proposed in [10]. Finally, S_i updates all myP_i and puts them into a **PInfoMsg**. After a lapse of $T_{forward}$, it broadcasts its own **PInfoMsg**.

Algorithm 3 Potential fields construction and update

1: **if** $State_i$ is **CLUSTER**(i,0) **then**
2: $netID_i \Leftarrow i$
3: $myP_i[i] \Leftarrow initial\_potential$
4: puts i into $pList_i$
5: puts $myP_i[i]$ into $vList_i$
6: broadcast **PInfoMsg**$(i, netID_i, pList_i, vList_i)$ per T_{flood}
7: clear $pList_i, vList_i$
8: **else**
9: $netID_i \Leftarrow NULL$
10: **end if**
11: **loop**
12: **if** receive **PInfoMsg**$(s, netID_s, pLIst_s, vLIst_s)$ **then**
13: **for** $j = 1$ to k **do**
14: update the **NTable**$(s, netID_s, pList_s[j], vList_s[j], t_{S_i})$
15: **end for**
16: **for all** $entry$ in **NTable do**
17: **if** $t_{S_i} - entry.time > T_h$ **then**
18: remove the $entry$ from **NTable**
19: **if NTable** has no $entry$ **then**
20: $State_i \Leftarrow$ **CLUSTER**$(i, 0)$
21: broadcast **StatePacket**
22: **else if NTable** has no $entry$ whose $entry.netID$ is $netID_i$ **then**
23: **if** $State_i$ is **CLUSTER**$(i, 0)$ **then**
24: $netID_i \Leftarrow i$
25: **else**
26: $netID_i \Leftarrow NULL$
27: **end if**
28: **end if**
29: **end if**
30: **end for**
31: **for** $j = 1$ to k **do**
32: update all $myP_i[pList_s[j]]$
33: **end for**
34: **if** $netID_i$ is $NULL$ or $netID_i > netID_s$ **then**
35: $netID_i \Leftarrow netID_s$
36: **end if**
37: **for** $j = 1$ to k **do**
38: puts $pList_s[j]$ into $pList_i$
39: puts $myP_i[pList_s[j]]$ into $vList_i$
40: **end for**
41: broadcast **PInfoMsg**$(i, netID_i, pList_i, vList_i)$ after $T_{forward}$
42: clear $pList_i, vList_i$
43: **end if**
44: **end loop**

C. Traveling to cluster heads according to potential fields

Each cluster head has a unique potential field and also has the minimum potential value in its potential field. All sensor nodes forward their sensing data to one of their neighbor nodes

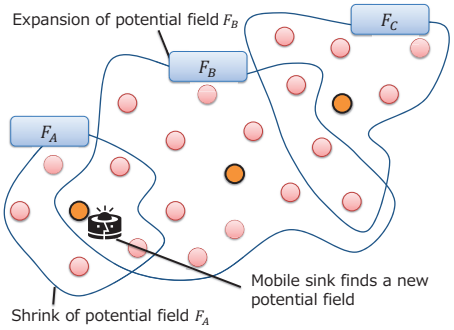

Figure 4. New potential field F_B is detected by the mobile sink after potential field F_A decreases its potential

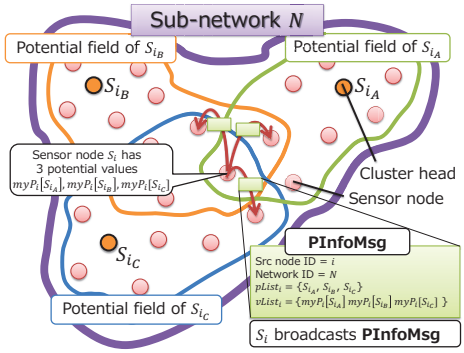

Figure 2. Example of a situation where S_i broadcasts a **PInfoMsg**

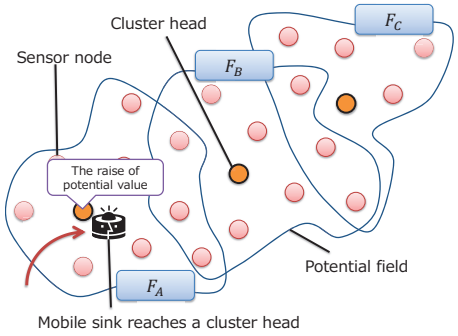

Figure 3. Increase of a potential value of a cluster head caused by arrival of the mobile sink

that have a smaller potential value, and then, all sensing data eventually reach one of the cluster heads. The mobile sink also utilizes these multiple potential fields, that is, utilizes $vList$ in a **PInfoMsg** transmitted by a sensor node to reach a cluster head [8]. After the arrival of the mobile sink at a cluster head, the cluster head gives collected data to the mobile sink and increases its potential value greatly. This decreases the priority of the potential field of the cluster head compared with other potential fields because a sensor node with a "smaller" potential value attracts data and the mobile sink as shown in Fig. 3. After that, the mobile sink can find a new potential field and can go to another cluster head as shown in Fig. 4.

When a mobile sink has visited all cluster heads in a sub-network, the mobile sink goes to an unvisited sub-network. Then, the mobile sink visits all sub-networks in the order that it previously visited and memorized them in the manner explained in Sec. II.

IV. Simulation Evaluation

In this section, we show that our proposal realizes reliable data collection even when additions or breakdowns of sensor nodes occur.

A. Simulation settings

We assume a 1,000 m $\times$ 1,000 m square region including the whole observed area and deploy N_s sensor nodes at uniformly random positions in the observed area. The communication range of a sensor node is represented by a circle with a radius of 100 m. Sensor nodes observe surrounding environmental phenomena and create a data packet that includes sensing data every one hour. Then, they forward the packet to one of the cluster heads. As an important assumption underlying reliable data collection, a retransmission algorithm can attain successful data transmission between two nodes and each node has sufficient memory not to drop any data.

In our simulation, randomly chosen N_b sensor nodes will fail, which causes other nodes to disconnect from the sub-network. Also, N_a sensor nodes will be added at random places in the observed area. Those failures and additions are taken at several pre-defined times.

A mobile sink, whose speed is set to 5 m/s, starts to move at 0 s and follows one of the proposed two mobility strategies. The mobility strategy of the mobile sink is being followed every one hour, and as the mobile sink finishes one strategy, it returns to the initial position to charge its battery. Here, the mobility strategy for learning sub-network positions is executed every LI hours.

We assume that the mobile sink can pass all the region of the observed area for evaluating basic performance. As an evaluation of reliable data collection, we calculate a data collection ratio (denoted by CR) every hour using (1).

$$CR = \frac{N_{CD}}{N_{ED}} \tag{1}$$

Here, N_{CD} and N_{ED} mean the number of collected and all generated data, respectively.

In our simulation, N_s, N_b, and N_a are set to 40, 5, and 10, respectively. Simulation time is set as 604,800 s (i.e., 1 week) and we performed 30 simulations with different sensor node positions for each LI ($LI = 1, 2, 3, 4, 5$).

B. Simulation results

Figure. 5 shows the transition of CR when LI is set to 1 and 5. In this figure, the Y-axis is the average of CR of 30 simulation trials. Even after node additions and failures occur, CR recovers from a temporary drop when LI is set to

 ISBN: 978-1-61208-391-9

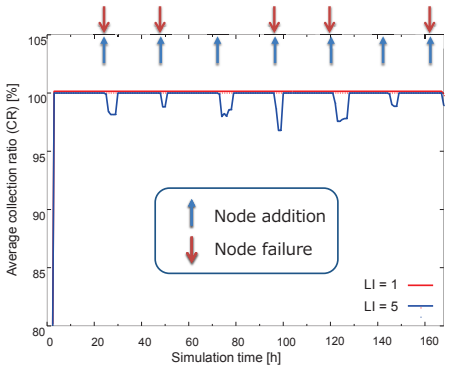

Figure 5. The transition of CR where $LI = 1$ and $LI = 5$

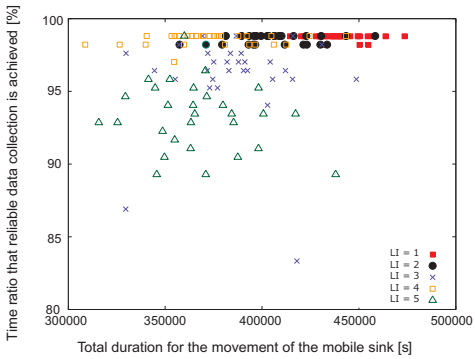

Figure 6. The relation between the reliability of data collection and the delay for mobility where $LI = 1, 2, 3, 4, 5$

5. This is because the mobile sink can learn the positions of new sub-networks every 5 hours. Thus, we find that reliable data collection can be attained by combining the two mobility strategies. Here, the average of CR remains 100% throughout the simulation when $LI = 1$. This is because the mobile sink always follows the strategy for learning the positions of all sub-networks.

Figure. 6 shows the relationship between the reliability and the efficiency of data collection. In this figure, the X-axis is the total time that the mobile sink moved except for waiting time at the initial position, and the Y-axis is the ratio of the time that the mobile sink can achieve 100% CR to the total simulation time.

The average durations for movement of the mobile sink are 438,428 s, 406,360 s, 385,653 s, 373,907 s, and 364,648 s, and the average time ratios that reliable data collection are achieved are 0.988, 0.986, 0.960, 0.986, and 0.934 for each $LI = 1, 2, 3, 4, 5$, respectively. The more frequent the mobile sink executes the mobility strategy for learning the sub-network positions, the more duration is required to acquire high reliability. Too frequent use of that strategy results in the redundant movements of the mobile sink in the case where few changes occur in the observed area. This trade-off relation between the reliability and the efficiency of data collection has to be managed with careful consideration of the frequency of environmental changes in the observed area.

V. Conclusion

In this paper, we present two mobility strategies for a mobile sink to realize reliable data collection. Our strategy can deal with a disconnection of a network and additional deployment of sensor nodes in an observed area. Through computer simulations, we demonstrate that reliable data collection is achieved by our proposal and show the trade-off between the reliability and the delay time for data collection. This trade-off can be adjusted by changing the frequency of the mobility strategy of the mobile sink for learning the positions of sub-networks. Our current interests are in the path planning among sub-networks, and in implementing our proposal in actual mobile nodes.

References

[1] H. Karl and A. Willing, *Protocols and architectures for wireless sensor networks*. Wiley, Oct. 2007.

[2] S. V. Deshpande and P. Devale, “Recent trends in using wireless sensor networks in industrial environment,” *International Journal of Computer Networking Wireless and Mobile Communication*, vol. 3, no. 3, pp. 11–20, Aug. 2013.

[3] J. Luo and J. Hubaux, “Joint sink mobility and routing to maximize the lifetime of wireless sensor networks: the case of constrained mobility,” *IEEE/ACM Transactions on Networking*, vol. 18, pp. 871–884, Jun. 2010.

[4] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas, “Sink mobility protocols for data collection in wireless sensor networks,” in *Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access*, Oct. 2006, pp. 52–59.

[5] Z. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, “Exploiting sink mobility for maximizing sensor networks lifetime,” in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Jan. 2005, pp. 287–295.

[6] F. Mourad, H. Chehade, H. Snoussi, F. Yalaoui, L. Amodeo, and C. Richard, “Controlled mobility sensor networks for target tracking using ant colony optimization,” *IEEE Transactions on Mobile Computing*, vol. 11, no. 8, pp. 1261–1273, Aug. 2012.

[7] R. Falcon, H. Liu, A. Nayak, and I. Stojmenovic, “Controlled straight mobility and energy-aware routing in robotic wireless sensor networks,” in *Proceedings of the 2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*, May 2012, pp. 150–157.

[8] S. Toyonaga, Y. Fujita, D. Kominami, and M. Murata, “Implementation of controlled sink mobility strategies with a gradient field in wireless sensor networks,” in *Proceedings of the 7th International Conference on Sensor Technologies and Applications*, Aug. 2013, pp. 27–32.

[9] 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, no. 5, pp. 780–790, May 2012.

[10] D. Kominami, M. Sugano, M. Murata, and T. Hatauchi, “Controlled and self-organized routing for large-scale wireless sensor networks,” *ACM Transactions on Sensor Networks*, vol. 10, p. 13, Nov. 2013.

 ISBN: 978-1-61208-391-9