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
†Graduate School of Economics, Osaka University, Japan
{y-fujita, murata}@ist.osaka-u.ac.jp
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, intrusion 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...
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.

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

1: // The mobile sink moves in every corner of the observed area.
2: repeat
3:   if intercepts PInfoMsg(i, netID_i, pList_i, vList_i) then
4:     if NetTable has no entry with netID_i then
5:       register NetTable(netID_i, pos, PInfoMsg, rssi)
6:     else
7:       if PInfoMsg.rssi > entry.rssi then
8:         update NetTable(netID_i, pos, PInfoMsg, rssi)
9:     end if
10:   end if
11: if State_i is CLUSTER(i, 0) then
12:     sends SensingDataRequest to S_i
13: end if
14: until reaches the end of the observed area

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_s</td>
<td>The number of sensor nodes which is initially deployed</td>
</tr>
<tr>
<td>N_f</td>
<td>The number of sensor nodes which will get failed</td>
</tr>
<tr>
<td>S_i</td>
<td>The sensor node whose ID is i</td>
</tr>
<tr>
<td>N_D(S_i)</td>
<td>The number of neighbor nodes of S_i</td>
</tr>
<tr>
<td>State_i</td>
<td>The state of S_i which indicates whether S_i belongs to a cluster or not.</td>
</tr>
<tr>
<td>t_Si</td>
<td>The current time of node S_i</td>
</tr>
<tr>
<td>T_limi</td>
<td>The time limit to search for its neighbor nodes</td>
</tr>
<tr>
<td>T_flood</td>
<td>The interval that cluster heads broadcast PInfoMsg</td>
</tr>
<tr>
<td>T_l</td>
<td>The period that NetTable keeps a non-updated entry</td>
</tr>
<tr>
<td>pID</td>
<td>The ID of the network that S_i belongs to</td>
</tr>
<tr>
<td>myP_i</td>
<td>The potential values that S_i has</td>
</tr>
<tr>
<td>pList_i</td>
<td>The set of pID that S_i has</td>
</tr>
<tr>
<td>vList_i</td>
<td>The set of myP that S_i has</td>
</tr>
</tbody>
</table>

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_limi). Then, S_i broadcasts a DegreePacket including ND(S_i), which is the number of received UpdatePackets until T_limi expires, to inform its node of its degree at t (T_limi ≤ t < T_limi + T_s_i). T_s_i is given from T_s_i = αe^{1/ND(S_i)} where α is a constant so that it ensures 0 < T_s_i ≪ T_limi. 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 UNCLUSTERED or CLUSTER(i, m) where m is larger than n + 1. S_i broadcasts a StatePacket if n + 1 ≤ 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
Algorithm 2 Selection of cluster heads in a network
1: // all nodes perform following;
2: State\textsubscript{i} \leftarrow UNCLUSTER
3: t \leftarrow \text{random value between } 0 \text{ and } T_{lim}
4: broadcast UpdatePacket at \textit{t\textsubscript{S}}
5: repeat
6: if receives a UpdatePacket then
7: \textit{N}(\textit{S}_1) \leftarrow \textit{N}(\textit{S}_1) + 1
8: end if
9: until \textit{t\textsubscript{S}} \geq T_{lim}
10: broadcast DegreePacket at \textit{t}, (T_{lim} \leq \textit{t} < T_{lim} + \textit{t\textsubscript{S}})
11: repeat
12: if receive StatePacket with CLUSTER(s, n) node
13: if State\textsubscript{ij} is UNCLUSTER then
14: State\textsubscript{ij} \leftarrow CLUSTER(s, n + 1)
15: else if State\textsubscript{ij} is CLUSTER(i, m) and m > n + 1 then
16: State\textsubscript{ij} \leftarrow CLUSTER(s, n + 1)
17: end if
18: if \textit{n+1} \leq \text{max} \textit{n} then
19: broadcast StatePacket
20: end if
21: until \textit{t\textsubscript{S}} \geq T_{lim} + \textit{t\textsubscript{S}}
23: if \textit{N}(\textit{S}_1) is larger than all neighbor nodes and State\textsubscript{i} is
24: UNCLUSTER then
25: broadcast StatePacket
26: end if

Algorithm 3 Potential fields construction and update
1: if State\textsubscript{i} is CLUSTER(i,0) then
2: netID\textsubscript{i} \leftarrow \textit{i}
3: \textit{myP}_i[i] \leftarrow initial potential
4: puts \textit{i} into pList\textsubscript{i}
5: puts \textit{myP}_i[i] into vList\textsubscript{i}
6: broadcast PInfoMsg(i, netID\textsubscript{i}, pList\textsubscript{i}, vList\textsubscript{i}) per \textit{T}\textsubscript{flood}
7: clear pList\textsubscript{i}, vList\textsubscript{i}
8: else
9: netID\textsubscript{i} \leftarrow NULL
10: end if
11: loop
12: if receive PInfoMsg(s, netID\textsubscript{s}, pList\textsubscript{s}, vList\textsubscript{s}) then
13: for \textit{j} = 1 to \textit{k} do
14: update the NTable(s, netID\textsubscript{s}, pList\textsubscript{s}[\textit{j}], vList\textsubscript{s}[\textit{j}], \textit{t}\textsubscript{S})
15: end for
16: for all entry in NTable do
17: if \textit{t\textsubscript{S}} - entry.time \textgreater T\textsubscript{h} then
18: remove the entry from NTable
19: if NTable has no entry then
20: State\textsubscript{i} \leftarrow CLUSTER(i, 0)
21: broadcast StatePacket
22: else if NTable has no entry whose
23: entry.netID is netID\textsubscript{i} then
24: if State\textsubscript{i} is CLUSTER(i,0) then
25: netID\textsubscript{i} \leftarrow \textit{i}
26: else
27: netID\textsubscript{i} \leftarrow NULL
28: end if
29: end if
30: end if
31: for \textit{j} = 1 to \textit{k} do
32: update all \textit{myP}_i[pList\textsubscript{s}[\textit{j}]]
33: end for
34: if netID\textsubscript{i} is NULL or netID\textsubscript{i} \textgreater netID\textsubscript{s} then
35: netID\textsubscript{i} \leftarrow netID\textsubscript{s}
36: end if
37: for \textit{j} = 1 to \textit{k} do
38: puts pList\textsubscript{s}[\textit{j}] into pList\textsubscript{i}
39: puts myP\textsubscript{i}[pList\textsubscript{s}[\textit{j}]] into vList\textsubscript{i}
40: end for
41: broadcast PInfoMsg(i, netID\textsubscript{i}, pList\textsubscript{i}, vList\textsubscript{i}) after \textit{T}\textsubscript{forward}
42: clear pList\textsubscript{i}, vList\textsubscript{i}
43: end if
44: end loop

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 1.

First, cluster head \textit{S\textsubscript{i}} initializes netID\textsubscript{i} and \textit{myP}_i, and
then broadcasts a PInfoMsg throughout the sub-network (lines 1–10). A PInfoMsg includes the sender’s ID (\textit{S\textsubscript{i}}), sub-network ID, multiple potential-field IDs (\textit{pID}), and potential values of correspondent potential fields as illustrated in Fig. 2. When receiving a PInfoMsg, a sensor node updates the NTable, \textit{myP}_i, and netID\textsubscript{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 (\textit{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. \textit{S\textsubscript{i}} registers or updates an entry of their NTable
when receiving a PInfoMsg and \textit{S\textsubscript{i}} removes an old entry that
is not updated for a period \textit{T\textsubscript{h}}. After that, \textit{S\textsubscript{i}} calculates its own
sub-network ID (netID\textsubscript{i}) and potentials (\textit{myP}_i). netID\textsubscript{i} is set
to the lowest value among \textit{pID} registered in its NTable and
\textit{myP}_i is set to an average potential value of its neighbors as
proposed in [10]. Finally, \textit{S\textsubscript{i}} updates all \textit{myP}_i and puts them into a PInfoMsg. After a lapse of \textit{T\textsubscript{forward}}, it broadcasts its

own PInfoMsg.

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).

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

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
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 duration for the movement of the mobile sink, and the Y-axis is the time ratio that reliable data collection is achieved. 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