Lightweight Scheduling of Data Collection Paths for Mobile Data Ferries in Long-term IoT Applications

You-Chiun Wang and Kuan-Chung Chen
Department of Computer Science and Engineering,
National Sun Yat-sen University, Kaohsiung, 80424, Taiwan
Email: ycwang@cse.nsysu.edu.tw; m043040017@student.nsysu.edu.tw

***Abstract*—Many IoT (Internet of things) applications rely on wireless sensors for long-term monitoring of the environment. Sensors need to relay their data to a sink via multihop transmissions, which consumes lots of energy and shortens their lifetime. Using a *mobile data ferry (MDF)* to visit sensors and collect their data can well address this issue. How to schedule a data collection path for the MDF is a challenge, and it is usually viewed as a variation of the *traveling salesman problem (TSP)*. However, some existing methods iteratively select a sensor for visiting and repeat TSP solutions many times, which incurs a high computation cost. Besides, they assume that sensors have large buffer space to cache data sent from others. By considering both computation overhead and buffer constraint, we develop a *lightweight and efficient path scheduling (LEPS)* scheme, whose idea is to select *delegate nodes (DNs)* along a routing tree of sensors by their relative distances and cached packets. Non-DN sensors send their data to the closest DNs, and the MDF then visits each DN to collect data. Through simulations, we show that LEPS spends less computation time to find a good data collection path such that sensors can save more energy and DNs better utilize their buffers to cache data, which verifies its effectiveness on long-term IoT applications.**

***Index Terms*—long-term monitoring, mobile data ferry, packet loss, path scheduling, traveling salesman problem.**

I. Introduction

Today, wireless sensor networks have been widely deployed to support a variety of applications, from home surveillance [1], [2] to smart buildings [3], intelligent vehicles [4], pollutant monitoring [5], [6], precision agriculture [7], grocery shopping [8], and health assessment [9]. They promote the development of *Internet of things (IoT)*, which allows objects (e.g., appliances or devices) forming a network to collect and transmit information by equipping with sensors or RFIDs [10]. To better offer IoT services, sensors should regularly report their sensing data for a long time [11]. However, sensors are powered by small batteries. Besides, it is not easy to replace batteries owing to the large number of sensors. Consequently, how to conserve energy of sensors on data transmissions (and prolong their lifetime) is a critical issue [12].

There are three common approaches to deal with the above issue. One is to adjust the duty cycle of each sensor, where sensors can sleep to save energy [13]. Another is to let sensors compress their sensing data, so as to decrease the amount of data transmissions [14]. The other is to use a *mobile data ferry (MDF)* to visit sensors and collect their data, so sensors need not relay their data via many hops [15]. However, the first two approaches may encounter the energy hole problem [16]. When a sensor is closer to the sink, it has to spend more energy on forwarding the data generated by more upstream sensors. Therefore, we aim at using the third approach to reduce energy consumption of sensors on data transmissions.

Since sensing data may have the delay requirement in IoT applications, how to schedule a short data collection path for the MDF to visit sensors is a challenge. Some methods [17], [18], [19] make the MDF call on each sensor to collect data, and the problem will be the *traveling salesman problem (TSP)*. Due to its NP-complete property [20], these methods can work well only with a small number of sensors. Given a large sensor network, it would be better to select a subset of sensors as *delegate nodes (DNs)* to cache data for others. Then, the MDF visits these DNs to receive data. Thus, the problem becomes how to find DNs such that sensors can spend the least energy on communications with the shortest data collection path.

In this paper, we propose a *lightweight and efficient path scheduling (LEPS)* scheme to select DNs and compute the MDF's path with less computation overhead. Given a routing tree of sensors, LEPS finds candidate DNs along the tree based on their available buffer space. Among the candidates, it iteratively selects DNs such that the distance between two adjacent DNs is reduced. Afterwards, LEPS finds the data collection path for the MDF to visit each DN through a TSP solution. Comparing with existing methods, our LEPS scheme has two novel designs. First, some methods seek to find out the shortest data collection path by repeatedly picking a sensor and using a TSP solution to check if it is suitable to serve as a DN. Thus, they will incur a high computation cost. On the contrary, LEPS selects DNs based on their relative distances and uses the TSP solution only once. In this way, we can greatly save the computation time. Second, existing methods assume that sensors generate the same number of packets (for sensing data) and have no limitation on buffer space. In practice, some DNs may need to cache data for more sensors to reduce the path. In this case, buffer overflow would occur on these DNs and the sink cannot get the lost sensing data. To deal with this issue, LEPS considers both available buffer space and the number of packets generated by each sensor when selecting DNs. Through simulations, we show that LEPS can decrease the computation time, save energy of sensors on communications, and prevent DNs from dropping packets. Therefore, it can better support long-term IoT applications than other methods.

II. Related Work

Some methods find the smallest set of DNs by restricting the hop count from each non-DN sensor to its closest DN. For example, [21] first organizes a shortest-path tree to link all sensors. From each leaf, the ancestor k-hops away is selected as a DN. Then, each DN collects data from all sensors in its subtree, and the MDF moves to visit DNs for data collection. Ma et al. [22] formulate a one-hop data collection problem to find the fewest DNs such that every non-DN sensor can directly forward its data to a one-hop DN. The problem is NP-hard, so they propose a spanning-tree covering heuristic to pick out DNs. In these methods, non-DN sensors can spend less energy on data transmissions, but each DN has to cache data for more sensors and spend more energy. Thus, some DNs may still encounter the energy hole problem.

Given a routing tree of sensors, [23] computes a weight for each tree edge by the number of sensors that use the edge to relay data. Then, it iteratively picks a DN based on edge weights and uses a TSP solution to modify the data collection path. Specifically, the TSP solution is run N_D times in a iteration, where N_D is the number of DNs. Thus, the scheme has time complexity of $O(N_D^2 \xi_{\text{TSP}})$, where ξ_{TSP} is the cost to run the TSP solution. The work of [24] proposes a *weighted rendezvous planning (WRP)* method by forming a spanning tree from sensors. Each sensor is given a weight $w_i = n_i \times h(i)$, where n_i is the number of packets that s_i sends to a DN and $h(i)$ is the hop count from the sensor to its nearest DN. Since WRP assumes that each sensor generates only one packet, n_i will be the number of the sensor's children plus one. Then, WRP selects a DN with the largest weight and uses a TSP solution to schedule a path to visit each DN. However, since it uses the TSP solution to recompute the path to visit DNs in every iteration, WRP has time complexity of $O(N^2 \xi_{\text{TSP}})$, where N is the number of total sensors.

Almiani et al. [25] propose a cluster-based scheme to select DNs, which arbitrarily picks a subset of sensors to be cluster heads and each of other sensors then joins the cluster whose head is the closest. Afterwards, the scheme selects one DN from each cluster, and calculates a data collection path to visit all DNs (i.e., by using a TSP solution). The above procedure is repeated until the path's length is above a threshold. However, since cluster heads are arbitrarily selected, some clusters may contain more sensors, which forces their sensors spending more energy on communications.

We can observe that the above methods run TSP solutions many times to amend the path, which incurs high computation overhead. In addition, none of them consider the limitation of buffer space. It thus motivates us to develop a lightweight scheme to find a data collection path for the MDF to collect data from sensors in a computation-efficient manner, which can be suitable for long-term IoT applications.

III. Problem Definition

We are given a wireless sensor network for the IoT application, which contains a set $\hat{S}$ of sensors and a sink s_0. Suppose that there exists a routing tree $\mathcal{T}$ (e.g., the shortest-path tree) to connect all sensors in $\hat{S}$ and s_0. In the IoT application, each sensor $s_i \in \hat{S}$ generates n_i packets of sensing data, where every packet has the same length of l bits. Initially, a sensor can cache no more than β packets in its buffer, where $\beta \in \mathbb{N}$. When a sensor s_i forwards one packet to its neighbor s_j, s_i has to spend an amount $(\alpha_T + \alpha_A \mathcal{L}^2(s_i, s_j)) \cdot l$ of energy [26], where α_T is the power taken by the transmitter circuit, α_A is the power taken by the amplifier circuit, and $\mathcal{L}(s_i, s_j)$ is the distance between s_i and s_j. On the other hand, s_j will spend an amount $\alpha_R \cdot l$ of energy to get the packet, where α_R is the power taken by the receiver circuit.

There is one MDF initially located at s_0. It will move to visit a set $\hat{D}$ of DNs to collect sensing data with a constant moving speed, where $\hat{D} \subseteq \hat{S} \cup \{s_0\}$. Besides, we assume that the communication time that the MDF spends to retrieve data from DNs can be ignored, as compared with its moving time. Then, our problem asks how to find the solution set $\hat{D}$ and compute a data collection path for the MDF to call on each DN in $\hat{D}$ and goes back to s_0 to offload its collected data, such that we can reduce the amount of communication energy spent by sensors and also decrease the number of packets dropped by DNs due to buffer overflow.

IV. The Proposed LEPS Scheme

In the LEPS scheme, each sensor s_i in $\hat{S}$ is associated with a flag f_i to indicate whether it has been designated as a DN or selected a DN to relay its data. Obviously, f_i is initially set to false for all sensors. Since the MDF starts its journey from s_0, we have $\hat{D} = \{s_0\}$ in the beginning. Then, LEPS contains the following three steps to find the data collection path:

- **Step 1: Search candidates.** We pick some sensors as the *beginning nodes* and add them to a set $\hat{B}$. Then, starting from each sensor in $\hat{B}$, we seek for candidates of DNs along the tree $\mathcal{T}$, and add these candidates to a set $\hat{P}$.
- **Step 2: Designate DNs.** Among potential candidates in $\hat{P}$, we select DNs based on their relative distances to the nodes in $\hat{D}$. Afterwards, we add these DNs to $\hat{D}$ and update $\hat{B}$ accordingly. In case that $\hat{B}$ becomes empty, which means that all DNs are found, we go to step 3. Otherwise, we go to step 1 to search for new candidates.
- **Step 3: Compute the path.** We then use a TSP solution to find a data collection path for the MDF to call on each DN in the solution set $\hat{D}$.

A. Step 1: Search Candidates

As the routing tree $\mathcal{T}$ decides the flow direction for sensors to send their data to the sink, we traverse $\mathcal{T}$ from leaves to the root. In particular, we add each sensor $s_i \in \hat{S}$ whose f_i is false to $\hat{B}$ if either s_i is a leaf or all children of s_i have false f_i values (i.e., they have been checked by LEPS).

Then, starting from each sensor in $\hat{B}$, we move towards $\mathcal{T}$'s root (i.e., s_0) and add up the number of packets generated by the visited sensors (denoted by A_i). When we visit a sensor s_i such that $A_i = \beta$ (or $A_k > \beta$ when visiting its parent s_k), which means that s_i is the last node which has enough buffer space to cache data for the visiting sensors, it will be a

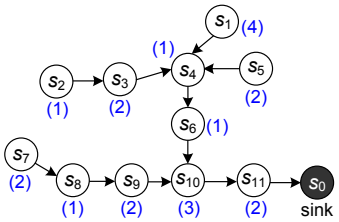

Fig. 1: Searching candidates in step 1.

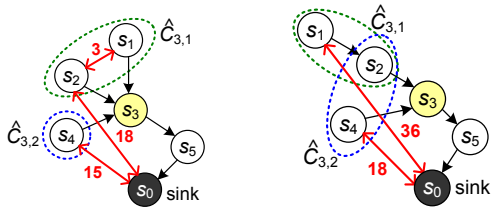

Fig. 2: Designating DNs in step 2: (a) $\hat{C}_{3,1} \cap \hat{C}_{3,2} = \emptyset$ and (b) $\hat{C}_{3,1} \cap \hat{C}_{3,2} = \{s_2\}$.

candidate of DN. In this case, we add s_i to $\hat{P}$. For convenience, let us denote by $\hat{C}_i$ the set of sensors that s_i should cache their data. Therefore, we can calculate that

$$A_i = n_i + \sum\nolimits_{s_j \in \hat{C}_i} n_j. \tag{1}$$

Each element of $\hat{P}$ is a two-tuple $(s_i, \hat{C}_i)$, so we can know the set of sensors that will send their data to a candidate s_i.

When we traverse $\mathcal{T}$, two cases should be considered for a visiting sensor s_i. In case 1, s_i has 0 or 1 child. When $A_i < \beta$, we then continue to check s_i's parent (i.e., s_k). However, if either $A_i = \beta$ or $f_k =$ false, s_i will be a candidate. In case 2, We use the *depth-first search (DFS)* to check s_i's children. Then, three conditions may occur:

- If the checking process terminates at any descendant, say, s_j of s_i (due to $A_i \geq \beta$), s_i should be a candidate.
- When we visit each child of s_i and get the same result of $A_i \geq \beta$, s_i will be a candidate.
- Otherwise, we use case 1 to continue the process.

We give an example in Fig. 1, where the number in each pair of parentheses indicates the number of packet generated by the corresponding sensor. In the example, β is set to 6 and we have $\hat{B} = \{s_1, s_2, s_5, s_7\}$. Then, step 1 is run as follows:

- From s_1: We first visit s_1 and s_4. By DFS, we then visit s_3 and find that $A_4 = n_1 + n_4 + n_3 > \beta$. Thus, s_4 is a candidate and $\hat{C}_4 = \{s_1\}$.
- From s_2: The visiting sequence is $s_2 \Rightarrow s_3 \Rightarrow s_4 \Rightarrow s_5$, so s_4 is a candidate and $\hat{C}_4 = \{s_2, s_3, s_5\}$.
- From s_5: The visiting sequence is $s_5 \Rightarrow s_4 \Rightarrow s_3 \Rightarrow s_2$, so s_4 is a candidate and $\hat{C}_4 = \{s_2, s_3, s_5\}$.
- From s_7: The visiting sequence is $s_7 \Rightarrow s_8 \Rightarrow s_9$, so s_9 is a candidate and $\hat{C}_9 = \{s_7, s_8\}$.

By combining the above results, we can calculate that $\hat{P} = \{(s_4, \{s_1\}), (s_4, \{s_2, s_3, s_5\}), (s_9, \{s_7, s_8\})\}$. Here, sensor s_4 is included in multiple elements of $\hat{P}$, each with different $\hat{C}_4$ sets. It means that s_4 can collect data from different sets of sensors, so these elements should be treated as different cases.

B. Step 2: Designate DNs

Then, each candidate s_i (i.e., the sensor indicated in the first tuple of each element in $\hat{P}$) is assigned with a weight by

$$W_i = \min\{h(s_i, s_j) \mid \forall s_j \in \hat{D}\}, \tag{2}$$

where $h(s_i, s_j)$ is the hop count from s_i to s_j on the routing tree $\mathcal{T}$. Here, W_i reflects the relative distance (i.e., in hop count) between s_i and its closest DN in $\hat{D}$. Since we traverse $\mathcal{T}$ from leaves to the root, we thus pick the candidate that has the largest weight (i.e., the most upstream one) as a DN. Fig. 1 shows an example, where $\hat{D} = \{s_0\}$. Since $W_4 = h(s_4, s_0) = 4$ and $W_9 = h(s_9, s_0) = 3$, we select s_4 to be a DN.

From the discussion in Section IV-A, a candidate s_i may have multiple $\hat{C}_i$ sets. However, s_i is allowed to choose only one $\hat{C}_i$ set to collect data, or its buffer must overflow. In this case, we should pick out "extra" DNs from other $\hat{C}_i$ sets accordingly. For ease of explanation, let us denote by s_i's $\hat{C}_i$ sets $\hat{C}_{i,1}, \hat{C}_{i,2}, \cdots, \hat{C}_{i,k}$. Then, two cases are discussed below.

Case 1: $\bigcap_{j=1}^{k} \hat{C}_{i,j} = \emptyset$. It means that any two $\hat{C}_{i,j}$ sets are disjointed. Thus, s_i can select one $\hat{C}_{i,j}$ set to collect data, and the child(ren) of s_i in each of other sets must serve as DN(s) to cache data for other sensors in the corresponding set. To do so, each $\hat{C}_{i,j}$ set is also assigned with a weight by

$$W_{i,j} = \min\{\mathcal{L}(s_j, \hat{C}'_{i,j}) \mid \forall s_j \in \hat{D}\} + \lambda(\hat{C}'_{i,j}), \tag{3}$$

where $\hat{C}'_{i,j}$ is the set of all s_i's children in $\hat{C}_{i,j}$, $\mathcal{L}(s_j, \hat{C}'_{i,j})$ is the shortest distance between a DN s_j and set $\hat{C}'_{i,j}$, and $\lambda(\hat{C}'_{i,j})$ is the length of a path to visit all sensors in $\hat{C}'_{i,j}$. Implicitly, when a $\hat{C}_{i,j}$ set has a larger weight, it means that the MDF has to move in a longer path to visit the DNs (i.e., s_i's children) in that set. Therefore, we will prefer letting s_i choose the $\hat{C}_{i,j}$ set with the largest weight, so as to reduce the data collection path of the MDF. Fig. 2(a) gives an example, where we select s_3 as a DN. In this example, we have $\hat{C}_{3,1} = \{s_1, s_2\}$ and $\hat{C}_{3,2} = \{s_4\}$, so their weights are $W_{3,1} = \mathcal{L}(s_0, s_2) + \mathcal{L}(s_2, s_1) = 21$ and $W_{3,2} = \mathcal{L}(s_0, s_4) = 15$, respectively. Therefore, s_3 should pick set $\hat{C}_{3,1}$ to collect data and s_4 will be a DN. In this way, we can reduce the MDF's path.

Case 2: $\bigcap_{j=1}^{k} \hat{C}_{i,j} \neq \emptyset$. Let us denote by $\hat{O}_{i,j}$ the set of overlapped sensors in $\hat{C}_{i,j}$ with others. In particular, no matter which $\hat{C}_{i,j}$ set s_i chooses to collect data, s_i can always get data from the sensors in $\hat{O}_{i,j}$. So, we need to select extra DNs from the sensors not in $\hat{O}_{i,j}$. Similarly, each $\hat{C}_{i,j}$ set is given a weight as follows:

$$W_{i,j} = \min\{\mathcal{L}(s_k, s_j) \mid \forall s_j \in \hat{D}\}, \tag{4}$$

where $s_k \in \hat{C}_{i,j} - \hat{O}_{i,j}$ such that $h(s_k, s_j)$ (i.e., the hop count between s_k and s_j) is minimized. Afterwards, s_i selects the $\hat{C}_{i,j}$ set with the largest $W_{i,j}$ value, and in each of other sets, we select the sensor s_k that does not belong to $\hat{O}_{i,j}$ and has

the smallest hop count to s_i to be a DN. Let us take Fig. 2(b) as an example, where s_3 is also selected as a DN. Thus, we have $\hat{C}_{3,1} = \{s_1, s_2\}$ and $\hat{C}_{3,2} = \{s_2, s_4\}$. Since $W_{3,1} = \mathcal{L}(s_1, s_0) = 36$ and $W_{3,2} = \mathcal{L}(s_4, s_0) = 18$, s_3 selects set $\hat{C}_{3,1}$ to collect data and s_4 becomes a DN accordingly.

After selecting s_i and some sensors in its $\hat{C}_{i,j}$ sets to be DNs, we then do the following actions: 1) add these sensors to the solution set $\hat{D}$, 2) mark the flag f_i of each of these sensors as true, 3) remove the elements whose candidate is s_i from $\hat{P}$, and 4) update $\hat{B}$ by the rules in step 1. Note that when $\hat{B} = \emptyset$, it means that all sensors in $\hat{S}$ have been checked, so we can go to step 3 to compute the data collection path. Otherwise, we return to step 1 to find out other candidates of DNs.

C. Step 3: Compute the Path

We use a TSP solution by local search [27] to find a data collection path for the MDF to visit each DNs in $\hat{D}$, whose time complexity is $O(|\hat{D}|^3)$. Fig. 1 gives an example to show how LEPS works, where $\beta = 6$. Let us denote by $\hat{F}$ the set of sensors whose f_i flags are false. Then, the example contains three iterations as follows:

1. $\hat{F}$ includes all sensors in $\hat{S}$, so $\hat{B}$ contains all leaves in $\mathcal{T}$ (i.e., s_1, s_2, s_5, and s_7). By step 1, we derive that $\hat{P} = \{(s_4, \{s_1\}), (s_4, \{s_2, s_3, s_5\}), (s_9, \{s_7, s_8\})\}$. Then, we select both s_1 and s_4 to be DNs. Therefore, we can get the result of $\hat{D} = \{s_0, s_1, s_4\}$.
2. Then, only sensors $s_6 \sim s_{11}$ remain in $\hat{F}$. Thus, we have $\hat{B} = \{s_6, s_7\}$ and $\hat{P} = \{(s_9, \{s_7, s_8\}), (s_{10}, \{s_6, s_9\})\}$. By step 2, we thus add s_9 to $\hat{D}$.
3. As $\hat{F} = \{s_6, s_{10}, s_{11}\}$, we obtain that $\hat{B} = \{s_6\}$. In this case, we compute that $\hat{P} = \{(s_{11}, \{s_6, s_{10}\})\}$. Although s_{11} should be selected as a DN, it actually can directly send its collected data to the sink s_0. Consequently, we need not add s_{11} to $\hat{D}$.

The solution set $\hat{D}$ includes s_0, s_1, s_4, and s_9. By the TSP solution, the path will be $s_0 \Rightarrow s_1 \Rightarrow s_4 \Rightarrow s_9 \Rightarrow s_0$.

V. Simulation Study

We develop a simulator in Java for performance evaluation. The sensing field is a square whose length is 200m, and there are 100–200 sensors deployed. A sensor has the communication distance of 20m and randomly generates 1 to 5 packets of sensing data, where the length of each packet is 134 bytes. We adopt the energy model in Section III, where α_T = 50 nJ/bit, α_A = 100 pJ/bit per m^2, and α_R = 50 nJ/bit. We compare LEPS with both WRP [24] and the cluster-based scheme [25], and apply the same local-search TSP solution [27] to them.

Fig. 3(a) gives the amount of computation time taken by each scheme. Since the WRP scheme repeatedly invokes the TSP solution whenever finding new DNs, its computation time grows exponentially when the number of sensors increases. The cluster-based scheme can reduce the computation time by grouping sensors into clusters and selecting one DN from each cluster. The computation time of our LEPS scheme grows very slowly as the number of sensors increases, since it uses the TSP solution only once after finding all DNs. In particular, when there are 200 sensors, the WRP and cluster-based schemes respectively require around 78.4 and 10.8 times of computation time than the LEPS scheme, which verifies that LEPS is lightweight and computation-efficient.

Fig. 3(b) shows the amount of sensors' energy consumed on communications. The WRP scheme seeks to optimize the data collection path of the MDF, so it achieves the lowest amount of energy consumption. Although our LEPS scheme is lightweight, it just slightly increases energy consumption of sensors. Specifically, LEPS increases no more than 8.3% of energy consumption than the WRP scheme. On the other hand, LEPS further reduces sensors' energy than the cluster-based scheme. In particular, sensors can save more than 10% of energy on communications in LEPS, as compared with the cluster-based scheme. This experiment demonstrates that our LEPS scheme can find a good data collection path to efficiently save the energy consumption of sensors, thereby extending their lifetime for long-term IoT applications.

Fig. 3(c) presents the number of packets dropped by DNs because of buffer overflow. Both the WRP and cluster-based schemes make DNs discard more packets when the number of sensors grows. Since the cluster-based scheme arbitrarily groups sensors into clusters to find DNs, some DNs may have to collect data from the sensors in larger clusters, thereby dropping more packets. On the contrary, our LEPS scheme considers available buffer space when selecting DNs, so it will not encounter packet loss on DNs. Moreover, LEPS allows DNs to better utilize their buffer to cache data. In particular, the average buffer utilization of DNs by the WRP, cluster-based, and LEPS schemes is 59.8%, 64.0%, and 74.4%, respectively, which shows that LEPS allows DNs caching more packets in their buffers than other schemes.

VI. Conclusion and Future Work

In long-term IoT applications, it is important to extend the lifetime of sensors by conserving their energy. Using an MDF to visit sensors and collect their data can help sensors save energy on communications, especially for those sensors close to the sink. This paper thus proposes the LEPS scheme to schedule the data collection path of the MDF. By considering buffer space of sensors and their relative distances, LEPS can efficiently pick out DNs and then use a TSP solution to compute the path in a short time. Through simulations, we show that LEPS is computation-efficient and can reduce communication energy of sensors while prevent DNs from discarding packets due to running out of buffer space, as comparing with both the WRP and cluster-based schemes.

We then discuss future directions. First, it is interesting to address transmission fairness of sensors [28] in the selection of DNs, especially when sensing data are heterogeneous [29]. Second, if sensors generate video streaming of sensing data [30], we have to consider their delay constraint. Third, we can apply data compression to DNs [31] to improve their efficiency of data collection. Finally, how to support secure data collection [32] deserves further investigation.

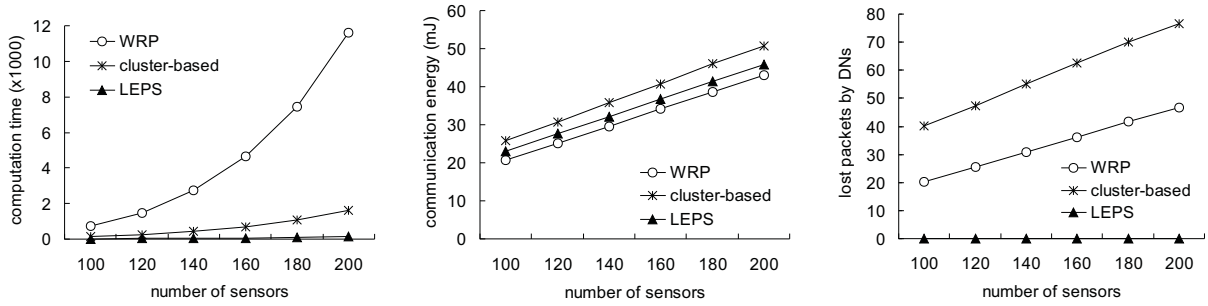

Fig. 3: Experimental results: (a) computation time, (b) energy consumption, and (c) packet loss.

ACKNOWLEDGEMENTS

You-Chiun Wang's research is co-sponsored by the Ministry of Science and Technology under Grant No. MOST 106-2221-E-110-022-MY2, Taiwan.

REFERENCES

[1] Y. C. Tseng, Y. C. Wang, K. Y. Cheng, and Y. Y. Hsieh, "iMouse: an integrated mobile surveillance and wireless sensor system," *IEEE Computer*, vol. 40, no. 6, pp. 60–66, 2007.

[2] Y. C. Wang, Y. F. Chen, and Y. C. Tseng, "Using rotatable and directional (R&D) sensors to achieve temporal coverage of objects and its surveillance application," *IEEE Transactions on Mobile Computing*, vol. 11, no. 8, pp. 1358–1371, 2012.

[3] L. W. Yeh, Y. C. Wang, and Y. C. Tseng, "iPower: an energy conservation system for intelligent buildings by wireless sensor networks," *International Journal of Sensor Networks*, vol. 5, no. 1, pp. 1–10, 2009.

[4] Y. C. Wang, "Mobile sensor networks: system hardware and dispatch software," *ACM Computing Surveys*, vol. 47, no. 1, pp. 12:1–12:36, 2014.

[5] S. C. Hu, Y. C. Wang, C. Y. Huang, and Y. C. Tseng, "Measuring air quality in city areas by vehicular wireless sensor networks," *Journal of Systems and Software*, vol. 84, no. 11, pp. 2005–2012, 2011.

[6] Y. C. Wang and G. W. Chen, "Efficient data gathering and estimation for metropolitan air quality monitoring by using vehicular sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7234–7248, 2017.

[7] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[8] Y. C. Wang and C. C. Yang, "3S-cart: a lightweight, interactive sensor-based cart for smart shopping in supermarkets," *IEEE Sensors Journal*, vol. 16, no. 17, pp. 6774–6781, 2016.

[9] M. Janidarmian, A. R. Fekr, K. Radecka, and Z. Zilic, "Multi-objective hierarchical classification using wearable sensors in a health application," *IEEE Sensors Journal*, vol. 17, no. 5, pp. 1421–1433, 2017.

[10] Y. C. Wang and S. J. Liu, "Minimum-cost deployment of adjustable readers to provide complete coverage of tags in RFID systems," *Journal of Systems and Software*, vol. 134, pp. 228–241, 2017.

[11] M. T. Lazarescu, "Design of a WSN platform for long-term environmental monitoring for IoT applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 1, pp. 45–54, 2013.

[12] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[13] J. Hao, B. Zhang, and H. T. Mouftah, "Routing protocols for duty cycled wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 116–123, 2012.

[14] Y. C. Wang, "Data compression techniques in wireless sensor networks," in *Pervasive Computing*. Nova Science Publishers, 2012.

[15] Y. C. Wang, F. J. Wu, and Y. C. Tseng, "Mobility management algorithms and applications for mobile sensor networks," *Wireless Communications and Mobile Computing*, vol. 12, no. 1, pp. 7–21, 2012.

[16] J. Ren, Y. Zhang, K. Zhang, A. Liu, J. Chen, and X. S. Shen, "Lifetime and energy hole evolution analysis in data-gathering wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 2, pp. 788–800, 2016.

[17] B. Yuan, M. Orlowska, and S. Sadiq, "On the optimal robot routing problem in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1252–1261, 2007.

[18] R. Sugihara and R. K. Gupta, "Optimal speed control of mobile node for data collection in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 1, pp. 127–139, 2010.

[19] L. He, J. Pan, and J. Xu, "A progressive approach to reducing data collection latency in wireless sensor networks with mobile elements," *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1308–1320, 2013.

[20] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics, 2007.

[21] M. Zhao and Y. Yang, "Bounded relay hop mobile data gathering in wireless sensor networks," *IEEE Transactions on Computers*, vol. 61, no. 2, pp. 265–277, 2012.

[22] M. Ma, Y. Yang, and M. Zhao, "Tour planning for mobile data-gathering mechanisms in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1472–1483, 2013.

[23] G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in wireless sensor networks with mobile elements," *IEEE Transactions on Mobile Computing*, vol. 7, no. 12, pp. 1430–1443, 2008.

[24] H. Salarian, K. W. Chin, and F. Naghdy, "An energy-efficient mobile-sink path selection strategy for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2407–2419, 2014.

[25] K. Almiani, A. Viglas, and L. Libman, "Tour and path planning methods for efficient data gathering using mobile elements," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 21, no. 1, pp. 11–25, 2016.

[26] Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, "Exploiting sink mobility for maximizing sensor networks lifetime," in *IEEE Annual Hawaii International Conference on System Sciences*, 2005, pp. 1–9.

[27] W. Zhang, "Depth-first branch-and-bound versus local search: a case study," in *National Conference on Artificial Intelligence*, 2000, pp. 930–935.

[28] Y. C. Wang, S. R. Ye, and Y. C. Tseng, "A fair scheduling algorithm with traffic classification in wireless networks," *Computer Communications*, vol. 28, no. 10, pp. 1225–1239, 2005.

[29] Y. C. Wang, "A two-phase dispatch heuristic to schedule the movement of multi-attribute mobile sensors in a hybrid wireless sensor network," *IEEE Transactions on Mobile Computing*, vol. 13, no. 4, pp. 709–722, 2014.

[30] W. H. Yang, Y. C. Wang, Y. C. Tseng, and B. S. P. Lin, "A request control scheme for data recovery in DVB-IPDC systems with spatial and temporal packet loss," *Wireless Communications and Mobile Computing*, vol. 13, no. 10, pp. 935–950, 2013.

[31] Y. C. Wang and C. T. Wei, "Lightweight, latency-aware routing for data compression in wireless sensor networks with heterogeneous traffics," *Wireless Communications and Mobile Computing*, vol. 16, no. 9, pp. 1035–1049, 2016.

[32] Y. C. Wang and Y. C. Tseng, "Attacks and defenses of routing mechanisms in ad hoc and sensor networks," in *Security in Sensor Networks*. CRC Press, 2006.