2011 Eighth IEEE International Conference on Mobile Ad-Hoc and Sensor Systems

Load Balanced Rendezvous Data Collection in Wireless Sensor Networks

Luo Mai[1], Longfei Shangguan[1], Chao Lang[1], Junzhao Du[1], Hui Liu[1], Zhenjiang Li[2,3], and Mo Li[3]
[1] Software Engineering Institute, Xidian University
[2] Department of Computer Science and Engineering, Hong Kong University of Science and Technology
[3] School of Computer Engineering, Nanyang Technological University
Email: {mailuo.cn, shanggdlk}@gmail.com, {dujz, liuhui}@xidian.edu.cn, lzjiang@cse.ust.hk, limo@ntu.edu.sg

***Abstract*—We study the rendezvous data collection problem for the mobile sink in wireless sensor networks. We introduce to jointly optimize trajectory planning for the mobile sink and workload balancing for the network. By doing so, the mobile sink is able to efficiently collect network-wide data within a given delay bound and the network can eliminate the energy bottleneck to dramatically prolong its lifetime. Such a joint optimization problem is shown to be NP-hard and we propose an approximation algorithm, named RPS-LB, to approach the optimal solution. In RPS-LB, according to observed properties of the median reference structure in the network, a series of Rendezvous Points (RPs) are selected to construct the trajectory for the mobile sink and the derived approximation ratio of RPS-LB guarantees that the formed trajectory is comparable with the optimal solution. The workload allocated to each RP is proven to be balanced mathematically. We then relax the assumption that mobile sink knows the location of each sensor node and present a localized, fully distributed version, RPS-LB-D, which largely improves the system applicability in practice. We verify the effectiveness of our proposals via extensive experiments.**

Keywords-wireless sensor networks, rendezvous data collection, mobile sink, network load balancing

I. INTRODUCTION

As a promising technology, Wireless Sensor Networks (WSNs) spawn a surge of previously unforeseen applications. The diversity of those emerging applications interprets its great success. One fundamental operation of such applications is **data collection**, which characterizes the transmission process of in-situ sensory data from sensor nodes to the base station over the network. A variety of critical applications and network operations, such as event detection [1], localization [2], network self diagnosis [3], network reconfiguration [4], robust message delivery [5], and etc., rely on data collection as a basic component.

In most of previous studies, the *static sink* was wildly adopted to conduct data collection in WSNs. Due to the multi-hop data transmission style, however, severely unbalanced energy consumption is caused with the node-to-sink traffic flow. Sensor nodes close to the sink node have to carry much more traffic overhead compared with distant sensor nodes. Since sensor nodes are highly restricted to the limited battery power supply, such unbalanced energy consumption results in the quick power depletion on part of the network, and dramatically shortens the lifetime of the network as a whole. To reduce the negative impact, recent research works introduce the *mobile sink* as a potential solution to the data collection problem. The mobile sink is usually a miniature vehicle or robot with the motion capability, which roams within the network, harvests sensory data at a series of intermediate Rendezvous Points (RPs), i.e., data collection positions, and carries harvested data back to the base station. Since the data collection positions are usually distributed across the entire network, the RPs implicitly average the traffic burden over the network and reduce the energy bottleneck in the network. The lifetime of the network can thus be significantly prolonged.

Compared with the traditional static data collection setting, data collection performed by the mobile sink is more complicated in the following two aspects: *mobile sink trajectory planning* and *network load balancing*. According to [6], the typical moving velocity of a mobile sink is around 0.1~2.0 m/s. It will lead to an extremely long data collection delay if the mobile sink visits a large portion of the network, which is normally unable to meet the delay requirement of many practical applications. As a matter of fact, the small moving velocity is the fundamental design restriction, since increasing the moving speed of the mobile sink will lead to a significantly increased manufacturing cost and energy consumption. For example, a Packbot node consumes about merely 60W when the moving speed is 1 m/s while the consumed energy increases quadratically with its speed as reported by [7]. On the other hand, the mobile sink collects only partial sensory data at every RP. Different from the scenario with the traditional static sink, only a *local* data routing tree is formed, rooted at each RP. All the local trees are not overlapped and jointly offer a full coverage of the entire network. Thus, the mobile sink can be guaranteed to collect the network-wide sensory data by visiting all RPs. In principle, the work loads of local routing trees rooted at RPs should be balanced. Note that with the required delay constraint, badly selected RPs and planned mobile sink trajectory may fail in collecting all sensory data across the network; and even worse, a trajectory path which optimizes the total energy cost over the network does not necessarily lead to balanced local routing trees. As a matter of fact, above two aspects need to be addressed together such that efficient data collection can be achieved and the network lifetime can be significantly prolonged at the same time. However, so far as we know, how to jointly design mobile sink trajectory planning and network load balancing is still not yet thoroughly investigated by the

978-0-7695-4469-4/11 $26.00 © 2011 IEEE
DOI 10.1109/MASS.2011.35

community, and we aim to systematically study such a joint optimization problem in this paper.

There have been initial attempts made to explore the data collection problem with mobile sinks. Most existing works, however, solely focus on the trajectory planning aspect. Without taking network load balancing into consideration, the produced local routing trees may be highly unbalanced and some of them may run out of energy rapidly, leaving other routing trees excessive residual energy. The lifetime of the network as a whole will be severely limited.

In this paper, we explore the possibility of combining mobile sink trajectory planning and network load balancing to jointly optimize data collection for the mobile sink in wireless sensor networks. The contributions of this paper can be summarized as follows. First, we formulate such a joint optimization problem as the Minimum-energy Rendezvous Point selection with Load Balancing (MRPLB) Problem and we prove it is NP-hard. Then, based on the observed properties from the median reference structure, we propose an approximation algorithm, RPS-LB, tailored for the trajectory planning with network load balancing consideration. Next, we mathematically prove the correctness of the proposed algorithm, analyze the algorithm performance, and derive its approximation ratio. To improve the applicability of RPS-LB, we relax the assumption that the location of each sensor node is known by the mobile sink and propose a localized, fully distributed algorithm, named RPS-LB-D, where the new RPs can be decided based on merely a part of the network knowledge. We verify the efficiency and effectiveness of our approaches via large-scale simulations.

The rest of this paper is organized as follows. Related work is reviewed in Section II and the problem is formulated in Section III. We specify the design detail of RPS-LB and prove its properties in Section IV. In Section V, the distributed realization of RPS-LB is presented. We evaluate the algorithms in Section VI and conclude the paper in Section VII.

II. Related Work

A surge of recent works exploit utilizing the sink mobility to reduce energy consumption in WSNs. [8] gives a survey on the usage of sink mobility for energy-efficient data collection in delay-tolerant WSNs. Chakrabarti et al. [9] show that, if actuators move along regular paths, sensors can predict their arrival after learning their movement pattern, which makes sensors free from detecting actuators' arrival by keeping monitoring the wireless communication channel. Several heuristics are proposed in [10][11] to schedule the movement of actuators such that source nodes can be visited according to their buffer limitation to avoid data loss. Wang et al. [12] show that constraining the mobile sink in the neighborhoods of a base station can maximize network lifetime. Shi et al. [13] designed a provably approximation algorithm regarding the location of a mobile base station in favor of maximizing network lifetime.

On the other hand, rendezvous-based data collection draws great attention recently by trading off the energy consumption and the data collection delay. In [14], sources send their sensory data to the nodes in the vicinity of actuator trajectories which are picked up as the actuators pass by. Rao et al. [15] presented a generic data collection framework without location information. However, these works do not focus on collecting sensory data within bounded time delay. Xing et al. [17] study rendezvous planning along a geometric tree that approximates the reporting tree of data sources. Furthermore, [16] studies the trade-off between the energy consumption and the time delay in the sensor networks. Recently, centralized techniques such as clustering [18][19] and intelligent algorithms [20] are also utilized to minimize the network energy consumption of relaying data from sources to several intermediate points. The key limitation of their works is the high dependence on the perfect network knowledge, imposing an unrealistic requirement to the mobile sink in terms of computation capacity and the memory size.

However, minimizing network energy consumption may not necessarily lead to the maximum network lifetime, since the energy consumption may not be evenly distributed. To the best of our knowledge, there is no existing work focusing on jointly optimizing both trajectory planning and work load balancing for data collection with the mobile sink in WSNs. By doing this, an efficient data collection can be achieved and the network lifetime can be prolonged at the same time.

III. Preliminary

In this section, we will formally introduce the problem we will discuss in this paper.

A. An Illustrative Example

In our problem context, a set of source nodes, i.e., sensors, that periodically generate sensory data at an equal rate, are deployed in the target field. The mobile sink roams in the network to collect all those sensory data by visiting a series of RPs, with a required data collection delay bound D. The delay bound can be measured by the maximum distance that the mobile sink is allowed to move. More precisely, the maximum length of the trajectory can be calculated by $L = v_{MS} D$, where v_{MS} is the average movement speed of the mobile sink. In this paper, our ultimate goal is to determine the location of each RP and a set of local routing trees rooted at those selected RPs such that *(a)* all sensory data can be collected within the given delay bound, *(b)* network load is evenly distributed across the network, and *(c)* sufficiently long lifetime of the network can be achieved. To better illustrate the considered problem, a simple example is given in Figure 1(a) and (b).

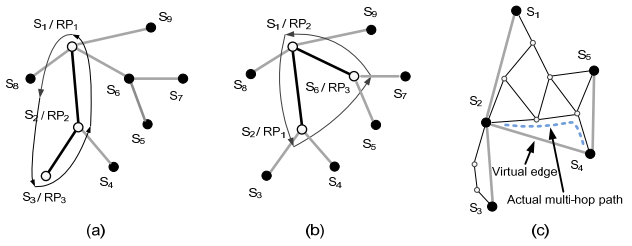

Figure 1 (a) Trajectory planning produced by the simple greedy strategy. (b) Trajectory planning with the workload balancing. (c) Underlying network topology and the geometrically approximated routing tree. The network topology is represented by black lines and hollow circles. The routing tree is represented by grey lines and solid circles.

Solid and hollow circles in Figure 1 represent source nodes and the RPs, respectively. Note that a RP can be a relay node as well. Suppose a greedy strategy is used, i.e., from an arbitrary source node, applying the depth-first search along any routing tree embedded in the network to explore a longest trajectory that satisfies the required delay bound. In Figure 1(a) the planed trajectory starts from source node S_1 and expands following the depth-first search path along the longest branch in the routing tree. Subject to the given delay bound constraint, the final trajectory cannot be further expanded after it reaches RP_3 as depicted in Figure 1(a). Such a simple strategy fails to satisfy three design requirements mentioned above, e.g., the size of the local routing tree rooted at RP_1 is much larger compared with routing trees at RP_2 and RP_3 (The local routing tree rooted at RP_3 includes source node S_3 only). As a result, the routing tree rooted at RP_1 needs to relay a much larger volume of sensory data and will become the energy bottleneck that limits network lifetime. Although the trajectory given in Figure 1(b) comprises the same number of RPs, its workload is much better balanced among different RPs, i.e., each RP maintains a local tree structure and relays sensory data for two source nodes. Consequently, there is not explicit energy bottleneck existing in any of the three local routing trees.

As the network grows, the problem becomes much more complicated and it is non-trivial to find the optimal trajectory with the balanced network load. In this paper, we refer to such a problem as the *Minimum-energy Rendezvous Point selection with Load Balancing (MRPLB) Problem*. We formally define the MRPLB problem in Section III.C.

B. Network Model

We assume that a set of sensor nodes $V = \{v_1, v_2, ..., v_n\}$ are randomly deployed in an $M \times M$ field. Nodes are assumed to know their physical location information and the data generation rates across the network are also equal. Location information can be obtained from the equipped GPS devices or underlying localization component. Set V contains both source nodes and relay nodes. Each source node $s_i \in S \subseteq V$ generates a certain amount of data at the beginning of collection period of D and data must be delivered to the mobile sink within D time. Such a delivery deadline is imposed by various reasons, such as the limited power supply of the mobile sink, the limited buffer size of sensors, or simply application requirements for data freshness. As mentioned before, the movement of the mobile sink is constrained by a given delay bound D as well, and this delay bound can be measured by the maximum distance that the mobile sink is allowed to move.

We assume that a logical routing tree T has been initially embedded in the network to connect all source nodes. Each edge on T represents a multi-hop path (via relay nodes). In most previous works [15][18], finding the optimal mobile sink trajectory always requires perfect network knowledge such as the topology of the entire routing tree and the locations of all source nodes and relay nodes; however, such information is expensive to be obtained in practice. Therefore, the global routing tree is a logical approximated tree that representing the geometrical features of actual network topology. Such concept is illustrated in Figure 1(c). The use of approximated tree allows us to determine the location of RPs without the global network information. Our algorithms can yield a better solution if the completed network topology is available.

To quantify the energy consumption of the proposed protocol, we assume that the total energy consumption of delivering a data packet along a path is proportional to the Euclidean distance between the source node and destination node. Such an assumption is usually valid when sensor nodes are densely deployed in the network. Nodes can approximately estimate their energy consumption during the data transmission based on the geographic relationship between any pair of source and destination nodes. Our work can also be extended to utilize the expected transmission count (ETX) as the link quality metric and its details can be found in our earlier work [24]. Plenty of existing data dissemination protocols, e.g. [16], also adopt such an energy model. In addition, we assume that the storage capacity of the mobile sink and sensor nodes is large enough to buffer the total volume of sensory data generated from source nodes within time D. Several recent sensor network platforms [21] can integrate 10~100 MB NAND flash memory with ultra-low power consumption.

C. Problem Statement

Now, we formally define the MRPLB problem as follows:

Definition 1: Given an initial routing tree $T(S, E)$ that connects a set of source nodes $S = \{s_i\} \subseteq V$ by edges E in the network, determine 1) a set of RPs $R = \{r_i\}$ and their sequence forming a trajectory U of the mobile sink that is no longer than $L = v_{MS} D$ and 2) a set of workload balanced routing trees covering all source nodes in S, such that $\min_{U} \sum_{s_i \in S} d_T(s_i, r_i)$, where $d_T(s_i, r_i)$ is the length of the path from s_i to its nearest r_i on tree T.

The energy consumption of each RP comes from receiving and transmitting sensory data within a period of D. In our network model, source nodes generate the same amount of sensory data within time D. The energy consumption of a specific RP r_i is therefore proportional to the number of its associated source nodes, which is defined as the *workload* of r_i. We focus on achieving workload *balancing* among RPs, i.e., every RP should be allocated almost the same number of source nodes. The optimization objective is to minimize the total energy consumption during the entire data collection process. Equivalently, it is to minimize the average energy consumption of each sensor node to prolong network lifetime.

The MRPLB problem in **Definition 1** can be proven NP-hard by the reduction from the Geometric Traveling Salesman Problem (G-TSP) [16]. The problem optimizes the locations of a set of RPs such that the network energy consumption incurred by data delivery can be minimized. In particular, a special-case decision version of MRPLB is to ask if there exists a set of RPs resulting in zero network energy consumption. Clearly, only when all source nodes are serving as the RPs, the network energy consumption can be zero, i.e., the mobile sink must visit every source via a tour within the limited length. It is exactly a decision version of the G-TSP problem, in which a salesman needs to visit a set of sites along a tour no longer than a given distance bound.

IV. PROTOCOL DESIGN AND ANALYSIS

A. Overview

The initial idea of our protocol is inspired by the following two observations that serve as the basic guidelines in designing algorithms in this section. Exiting state-of-the-art rendezvous-based data collection protocols, such as [18][19][20], attempt to provide a one-time solution for addressing the routing structure formation problem with a set of networking constraints, which usually incurs high computation and communication costs, suffers a relatively longer delay, and may rely on perfect network knowledge. Different from existing solutions, we plan to determine the optimal location of each RP through an incremental process. In particular, we start from selecting a sensor node as the *reference node* that can balance the current workload of the network. We incrementally expand the RPs set while keeping the mobile sink trajectory within the required delay bound. During the entire incremental trajectory planning process, the balanced workload structure should be maintained in our proposed solution.

The balanced workload at the reference node indicates the *benefit* to deploy the first RP close to the reference node. During the incremental trajectory planning, the benefit needs to be updated once a new RP is added, which requires maintaining a *reference structure* to guide the deployment of subsequent RPs. With such a reference structure, the overall energy consumption of the network could be reduced. At the same time, workload balancing among different local routing trees at RPs should be achieved as well. In addition, the reference structure needs to guarantee that the length of the trajectory formed by all deployed RPs is bounded by the maximum moving distance L, imposed by the delay bound D.

B. Median Searching Algorithm Design

Essentially, the reference structure resides at the *median* of the global routing tree and locations of all RPs can be determined if the median is founded. The median is a node on the tree. It not only minimizes the total energy consumption of gathering sensory data at itself, but balances the current network load in an optimal manner as well. Thus, the first step in our protocol is to efficiently locate the median on a routing tree in the network. Peng *et al.* propose an algorithm to find the median of a tree in [22]; however, the length of each edge in [22] is required to be identical. Since each edge in our geometrically approximated routing tree may represent a multi-hop path, whose length is proportional to its Euclidean distance. As a result, such an existing algorithm cannot be applied to our scenario directly.

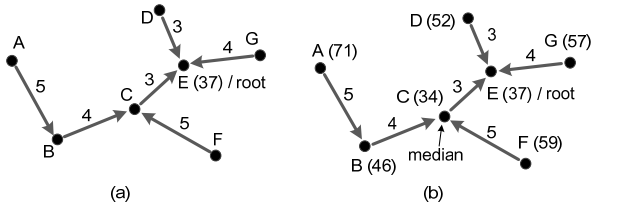

Figure 2 Execution of Algorithm 1. The number close to each edge indicates its length in Euclidean distance. (a) Orient the routing tree T into a directed rooted tree. (b) Determine the median.

Median Searching Algorithm

Input: routing tree T

Output: median m

1) Arbitrarily select a node r as the root of the inputted routing tree T and orient it into a rooted directed tree T_r. For any node v on T_r, we compute the $|T_v|$.
2) Traverse the tree T_r in a bottom-up manner (from leaves to the root) to compute the $D_{T_r}(r)$ via the formula in **Lemma 1**.
3) Compute the $D_{T_r}(v)$ for any node v on T_r in breadth-first fashion using the formula in **Lemma 2**.
4) The median m is the node with the minimum $D_{T_r}(.)$.

Algorithm 1 The pseudo code of the median searching algorithm.

To this end, we design a median searching algorithm to locate the median in $O(|S|)$ time. The efficient searching process depends on two lemmas as follows, which enable us to decide the median by simply traversing sensor nodes in the routing tree in a bottom-up, breadth-first search manner.

Lemma 1: Given a directed minimum spanning tree[1] T_r rooted at any source node r, we can compute the sum of distances from all other source nodes in T_r to node r in a bottom-up manner via the formula:

$$D_{T_r}(r) = \sum_{v \in desc(r)} \left(|T_v| \bullet d_{T_r}\left(v, par(v)\right)\right),$$

where *desc(v)* denotes the descendent of node v, *par(v)* denotes the parent of node v, and $|T_v|$ denotes the number of nodes in the local routing tree rooted at v.

Lemma 2: Given a directed minimum spanning tree T_r rooted at any node r, we can compute the sum of distances from all other nodes in T_r to an arbitrary node v in breadth-first fashion via the formula:

$$D_{T_r}(v) = D_{T_r}\left(par(v)\right) + (|T_r| - |T_v|) \bullet d_{T_r}(r,v) - |T_v| \bullet d_{T_r}(r,v).$$

The pseudo code of the Median Searching algorithm is given in Algorithm 1 and we also provide a simple example in Figure 2 to illustrate the algorithm. The algorithm contains four major steps. At *step (1)*, we randomly select a node, e.g., node E in Figure 2, as the root node. According to the formula given in Lemma 1, we get that $D_{T_E}(E)$ is 37. Then at *step (3)*, $D_{T_E}(.)$ for each source node can be calculated in breadth-first fashion using the formula given in Lemma 2. For instance, we can first obtain the values of $D_{T_E}(D) = 52$, $D_{T_E}(G) = 57$, $D_{T_E}(C) = 34$. Next, $D_{T_E}(B)$ and $D_{T_E}(F)$ can be calculated. In the end, $D_{T_E}(A)$ is determined to be 71. Clearly, $D_{T_E}(C)$ is the one with the minimum $D_{T_E}(.)$ value compared with all other nodes. Thus node C is selected as the median in this example.

C. RPS-LB Algorithm Design

Based on the obtained median, we introduce the design detail of *Rendezvous Points Selection with Load Balancing* (RPS-LB) algorithm in this subsection. Notice that after the

[1] Note that the reference structure can be formed based on the Minimum Spanning Tree (MST) rooted at any source node in the network. MST can be efficiently built up by the well-known *Kruskal* algorithm. For the presentation simplicity, we assume that there exists a global MST embedded in the network initially and our protocol will run on top of this global routing tree.

median is determined, the median becomes the initial position to form and update the reference structure in the network and we name such a structure as the *median reference structure* in the rest of this paper. Such structure is a subtree on the routing tree in actual. It not only minimizes the total distances from all sources to itself, but minimizes the largest branch it incurs. Implicitly, such structure can be exploited to guide us in finding the optimal location for each RP. The median reference structure has several important properties related to our design as follows: *(a)* The total energy consumption of transmitting sensory data from sources nodes to the median reference structure is monotonically decreasing when its size increases; (*b)* The number of nodes in the largest local routing tree monotonically decreases when its size increases; and (*c)* The size of the median reference structure can implicitly indicate the length of the resulting trajectory planned for the mobile sink. Such properties inspire us that deploying RPs at the intersections between the median reference structure and the approximated routing tree may yield a good solution for trajectory planning with the workload balancing. We will later show that the argument turns out to be true.

RPS-LB operates iteratively. In each iteration, the current trajectory of the mobile sink is expanded by adding a new RP to share the load of the RP with the heaviest workload. In addition, to satisfy the maximum moving distance requirement, the quantity of the median reference structure expansion in each iteration should be restricted as well. Such a process allows RPS-LB to dynamically migrate the workload of source nodes with heavy traffic burden to those with light traffic burden, and thus achieves well planned trajectory with balanced workload. Algorithm 2 shows the pseudo code of RPS-LB, in which δ is a parameter set by the system operator according to the desirable trade-off between the solution quality and the computational complexity. When δ is small, RPS-LB operates with more iterations yet provides more RP candidates. Note that the *size* of a tree in this subsection is defined as its total edge length along the tree.

Figure 3 illustrates how the RPS-LB algorithm works. For simplicity, we omit the details of the branches B_1, B_2, B_3, and B_4, where the numbers of nodes belonging to B_1, B_2, B_3, and B_4 are assumed to be equal. At *step (1)* of Algorithm 2 the median reference structure T_m (represented by the black dotted line segments) only contains the medians. From *step (2)* to *step (3)*, RPS-LB expands T_m toward the largest branch(s) with an equal rate (We set the *rate* as a constant value, i.e., 1 *m* per unit time). The intermediate result is shown in Figure 3(a), where $R = \{s_2, s_5, s_6, s_3\}$ and they are all the intersections between T_m and the paths from source nodes to medians. For example, the RP r_4 is the intersection node between T_m and the path $s_4 \rightarrow m_1$. Then, at *step (5)* and *step (6)*, RPS-LB checks whether T_m can be further expanded due to the delay bound *D*. If L -TSP(R) $> \delta$, **Theorem 3** guarantees that the answer is positive and we will prove it soon. *TSP*(R) is a TSP solver that returns the length of a sink tour that visit all the RPs in R. After the first iteration, RPS-LB finds that T_m can be further expanded. Thus, the algorithm executes back to *step (2)* and further grows T_m towards the current largest branches $\{s_1, B_1\}$, $\{s_4, B_2\}$, $\{s_7, B_3\}$, $\{s_8, B_4\}$, as shown in Figure 3(b). After the second iteration, RPS-LB finds that the size of T_m reaches *Y*. In other words, L -TSP(R) $< \delta$ and RPS-LB returns R.

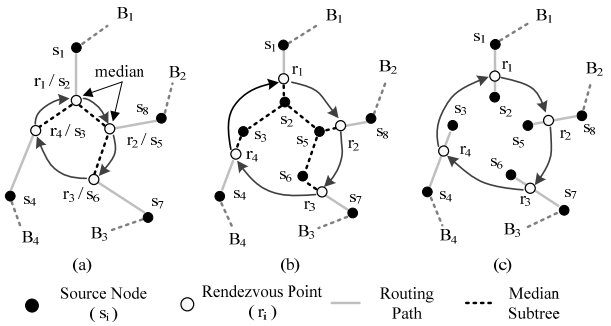

Figure 3 An example of the RPS-LB algorithm's execution. (a) T_m after the first iteration. (b) The final T_m is of size *Y*. (c) The final solution.

RPS-LB Algorithm

Input: routing tree $T(S,E)$, median *m,* sink tour length *L,* and δ

Output : The RPs set R

1) Let $Y = L / 2$, $T_m = \{m\}$. // T_m is the median reference structure
2) Count the number of nodes in each branch in the set $\{T - T_m\}$. Denote the branches with the maximum nodes as B_{max}.
3) T_m grows into each branch in B_{max} at an equal rate (constant value) until the size of T_m reaches Y or any node not in T_m is included.
4) $R = \{r_i \mid r_i$ is the intersection node between T_m and the path $s_i \rightarrow m$ and $s_i \in S\}$.
5) **if** $X = L - \text{TSP}(R) > \delta$
6) $Y = Y + X / 2$; **goto step** 2);
7) **else return** R.

Algorithm 2 The pseudo code of the RPS-LB algorithm.

To construct a final solution, the nodes residing in the interior of median reference structure are assigned to their nearest RPs respectively, and the final result is shown in Figure 3(c).

RPs found by the RPS-LB algorithm are represented by the physical locations. It is possible that there is no sensor nodes residing exactly at those locations to serve as RPs. We address this issue as follows: the mobile sink sends out an Anycast message [23] when arriving at the desired position. Sensor nodes around the mobile sink will receive such a message and respond. The mobile sink will select the first responding sensor node as the RP.

D. Theoretical Analysis

In the following, we first prove the correctness of our proposed RPS-LB algorithm, and then derive its approximation ratio to show its effectiveness. In this subsection, for a given graph G, we define the *size* of G, denoted by $s(G)$, as the total lengths of the edges on it.

Theorem 1: Among all possible subtrees of size *Y*, the total of distances from all source nodes in the routing tree to the median subtree[2] is always minimized.

Proof: We omit the proof due to the page limitation.

[2] From the example given in Figure 3 we can see that the median reference structure is essentially a tree structure. Actually, it is true in general. We omit the proof of this fact due to the page limitation and use **median reference structure** and **median subtree** interchangeably in the rest of this paper.

Remarks: As RPs are deployed at the intersection between the median subtree and the routing tree, Theorem 1, minimizing the distance-sum of the median subtree, ensures that the network operated by RPS-LB experiences the minimum energy consumption of relaying sensory data from source nodes to the RPs compared with other protocols.

Theorem 2: Among all possible subtrees of size *Y,* the number of nodes in the largest branch induced by the median subtree is always minimized.

Proof: We omit the proof due to the page limitation.

Remarks: The RPs collect sensory data from the sources in their local routing trees, i.e. their associated branches. It is clear that the workload will be balanced if the size of the largest branch is small in the network. Theorem 3 shows that our proposed RPS-LB can achieve such a goal.

Consequently, Theorems 1 and 2 jointly demonstrate the workload can be well balanced among the RPs during trajectory planning based on our proposed algorithm. Next, we prove that there always exists a TSP tour no longer than *L* that allows the mobile sink to visit all the RPs in *R* (determined by Algorithm 2) within the delay bound.

Theorem 3: $TSP(R) < L$, i.e., *step (5)* in RPS-LB, always holds before RPS-LB's termination, where R_i is the RPs set found in the *i*-time iteration.

Proof: We omit the proof due to the page limitation.

Remarks: Theorem 3 shows that there always exists a trajectory no longer than *L* to visit all selected RPs. This enables the output solution to satisfy the delay constraint in our network model.

Now we focus on deriving the approximation ratio of the RPS-LB algorithm to deeply understand how close the proposed algorithm can perform compared to the optimal solution, which characterizes the performance quality of the trajectory planned by RPS-LB. Suppose MST_S is the minimum spanning tree connecting source nodes set *S*. Let β be the ratio of *L* to the total edge length of MST_S, i.e., $\beta = L/s(MST_S)$. We assume that $\beta \ll 1$, because if *L* is too long, the data collection delay becomes extremely large due to the mobile sink's low movement speed. In addition, the power supply of the mobile sink may not support such a long-distance traversal in many real applications.

We define the *e-distance* between node *u* and *v*, denoted by $e_T(u,v)$, as the length of the path $u \to r' \to v$ on tree *T*, where r' is the nearest RP to *u*. Such distance metric can be paraphrased as the length of data delivery path, where source nodes should first send their sensory data to the respective nearest RPs and then proceed to the destination node. For instance, in Figure 3(b), $e_T(s_6, s_5)$ is equal to the length of path $s_6 \to r_3 \to s_6 \to s_5$. Let $E_T(S,u)$ represent the *e-distance-sum* from all source nodes to node *u*, i.e., $E_T(S,u) = \sum_{v \in S} e_T(v,u)$. The use of *e-distance* facilitates our expression of the network energy consumption incurred by the solution from RPS-LB. Besides, we also define *cluster(s)* of the RP node *r*, denoted by *c*(*r*), as the set of branches all rooted at *r*. For instance, in Figure 3(c), $c(r_2) = \{\{s_5\}, \{s_8, B_2\}\}$.

Theorem 4: The approximation ratio of RPS-LB is no greater than $1+\lambda$, where $\lambda = \beta / (1-\beta \cdot (\alpha - 1/2))$ and $\alpha = |S|/2$, $\beta = L/s(MST_S)$ and $\beta \ll 1$.

Proof: Suppose R^* represents the set of the RPs in the optimal solution. T^* represents the set of the local routing trees rooted at the R^*. We first derive the energy consumption of the optimal solution to be the lower bound of our proposal. As the energy cost of transmitting a data packet is proportional to the Euclidean distance between the sender and receiver, the total consumption of transmitting data is proportional to the total of distances from every source $s \in S$ to its $r_i \in R^*$ following the corresponding $t_i \in T^*$. This cost can be represented by

$$C_{opt} = \kappa \bullet D_{T^*}(S, R^*) \tag{5-1}$$

where $D_{T^*}(S,R^*) = \sum_{s_i \in S,\ r_i \in R^*} d_{T^*}(s_i, r_i)$, $d_{T^*}(s_i, r_i)$ represents the distance of routing path from source s_i to its RP r_i on the tree $t \in T^*$, and the constant κ denotes the energy cost that is required to transmit a packet ahead on its way per meter.

Let MST_{R^*} denote the minimum spanning tree with the terminal nodes set as R^*. Note that MST_{R^*} is a subtree reside in the interior of the abovementioned MST_S. According to the definition of minimum spanning tree, the total edge length of the union of T^* and MST_{R^*} is no shorter than the total edge length of MST_S. Hence,

$$s(T^*) + s(MST_{R^*}) \ge s(MST_S) \tag{5-2}$$

As the paths connecting sources and the RPs are overlapped with each other, $D_{T^*}(S,R^*)$ would never be smaller than the total edge length of T^*. Thus, the equation (5-2) can be transformed to:

$$D_{T^*}(S,R^*) + s(MST_{R^*}) \ge s(MST_S) \tag{5-3}$$

Let T_{R^*} denote a subtree induced by removing an edge from the TSP tour $TSP(R^*)$ that visits each node in R^*. It is obvious that $s(T_{R^*}) < TSP(R^*)$. Then, we have

$$D_{T^*}(S,R^*) \ge s(MST_S) - s(MST_{R^*}) \ge s(MST_S) - L \tag{5-4}$$

According to the definition of e-distance, only in the case that the destination sets are the same RPs set, the e-distance-sum and distance-sum both deriving from the same sources set can be equal, i.e., $D_{T^*}(S,R^*) = E_{T^*}(S,R^*)$. Hence, (5-4) can be further transformed into:

$$E_{T^*}(S,R^*) \ge s(MST_S) - L \tag{5-5}$$

where $E_{MST_S}(S,m)$ represents the e-distance-sum from all source nodes in *S* to the median node *m* through the tree MST_S. According to the definition of median, any path starting from a RP and ending at the *m* is no longer than $L/2$, where r_i denotes the ith node in the optimal RPs set R^*, i.e., $e_{MST_S}(r_i \to m) = d_{MST_S}(r_i \to m) \le L/2$. Hence,

$$\begin{aligned} E_{MST_S}(S,m) &= E(S,R^*) + \sum_{r_i \in R^*} |c(r_i)| \bullet e_{MST_S}(r_i \to M) \\ &\le E_{MST_S}(S,R^*) + \sum_{r_i \in R^*} |c(r_i)| \bullet L/2 \\ &\le E_{MST_S}(S,R^*) + |S| \bullet L/2 \end{aligned} \tag{5-6}$$

If $\alpha = |S|/2$, we further have:

$$E_{MST_S}(S,m) \le E_{MST_S}(S,R^*) + \alpha \cdot L \tag{5-7}$$

In the following, we proceed to discuss the energy consumption of data transmission incurred by the RPS-LB

algorithm. For simplicity, we first analyze the energy cost incurred by a RP solely. Such cost, denoted by C_{r_i}, can be represented as follows:

$$C_{r_i} = \kappa \bullet \left(E_{MST_S}\left(c(r_i), m\right) - \left|c(r_i)\right| \bullet E_{MST_S}\left(r_i, m\right)\right) \quad (5\text{-}8)$$

Through summing up the equation (5-8), we can obtain the total energy consumption C_{RPS-LB}:

$$\begin{aligned} C_{RPS-LB} &= \sum_{r_i \in R} C_{r_i} \\ &= \sum_{r_i \in R} \kappa \bullet \left(E_{MST_S}\left(c(r_i), m\right) - \left|c(r_i)\right| \bullet E_{MST_S}\left(r_i, m\right)\right) \\ &= \kappa \bullet \left(\sum_{r_i \in R} E_{MST_S}\left(c(r_i), m\right) - \sum_{r_i \in R} \left|c(r_i)\right| \bullet E_{MST_S}\left(r_i, m\right)\right) \\ &= \kappa \bullet \left(E_{MST_S}(S, m) - \sum_{r_i \in R} \left|c(r_i)\right| \bullet E_{MST_S}\left(r_i, M\right)\right) \end{aligned} \quad (5\text{-}9)$$

According to the definition of the RPs in the RPS-LB algorithm, there is at least one source associated with each RP. Based on this observation, we can know $|c(r)| \geq 1$. Besides, the execution of RPS-LB can always guarantee that $\sum_{r_i \in R} E_{MST_S}(r_i, m) \geq s(T_m) \geq L/2$, where T_m is the median subtree. In short, we have

$$\begin{aligned} &\sum_{r_i \in R} |c(r_i)| \bullet E_{MST_S}(r_i, m) \\ &\qquad \geq \sum_{r_i \in R} E_{MST_S}(r_i, m) \geq L/2 \end{aligned} \quad (5\text{-}10)$$

In addition, let $L = \beta \bullet s(MST_S)$ to be integrated with (5-5), we have

$$L \leq \frac{\beta}{1-\beta} \bullet E_{MST_S}(S, R^*) \quad (5\text{-}11)$$

Further integrating the equation (5-10) and (5-11) with (5-7), then

$$\begin{aligned} E_{MST_S}\left(S, R^*\right) &\geq E_{MST_S}\left(S, m\right) - \alpha \bullet L \\ &= E_{MST_S}\left(S, m\right) - L/2 - \left(\alpha - 1/2\right) \bullet L \\ &\geq E_{MST_S}\left(S, m\right) - \sum_{r_i \in R} |c(r_i)| \bullet E_{MST_S}\left(r_i, M\right) \\ &\qquad - \left(\alpha - 1/2\right) \bullet L \end{aligned} \quad (5\text{-}12)$$

Multiplying the constant κ with both sides of (5-12), and simultaneously integrating with the equation (5-1) and (5-9), we can finally establish the relationship of energy cost between the optimal solution and the RPS-LB algorithm,

$$\begin{aligned} \kappa \bullet \left(E_{MST_S}\left(S, m\right) - \sum_{r_i \in R} \left|c(r_i)\right| \bullet E_{MST_S}\left(r_i, m\right)\right) &\leq \kappa \bullet \left(E_{MST_S}\left(S, R^*\right) + \left(\alpha - 1/2\right) \bullet L\right) \\ C_{RPS-LB} &\leq C_{opt} + \frac{\beta}{1-\beta} \bullet \left(\alpha - 1/2\right) \bullet C_{opt} \\ C_{RPS-LB} &\leq \left(1 + \frac{\beta}{1-\beta} \bullet \left(\alpha - 1/2\right)\right) \bullet C_{opt} \end{aligned} \quad (5\text{-}13)$$

Remarks: If the delay bound is extremely small or the energy supply of the mobile sink is highly limited, the coefficient β becomes very small and the derived approximation ratio indicates that the performance of RPS-LB is close to the optimal solution.

V. LOCALIZED PROTOCOL DESIGN

As mentioned before, the proposed RPS-LB relies on the location information of each source node. Such global information, however, usually limits the scalability of the system and hinders the applicability of the proposed protocol. To enhance the applicability of RPS-LB in practice, we release the requirement about the perfect location information at the mobile sink side and propose a localized, fully distributed version of the protocol named as RPS-LB-D in this section.

In Algorithm 2, RPS-LB explores a new RP only depending on the current workload of each RP already deployed, which inspires us that if such a decision can be made based on merely local information, the global network knowledge (such as the topology of the approximated routing tree) required in RPS-LB can be avoided. Based on our study, we find that such a goal can be achieved in practice. In the network, the mobile sink always knows the size of the local routing tree rooted at each RP, based on which the mobile sink is aware where the current energy bottleneck is. Such local information is enough for the mobile sink to decide how to expand its current trajectory in local. We implement such an idea into RPS-LB-D and describe it in detail in the rest of this section.

A. Network Initialization

The RPS-LB-D algorithm starts with a two-phase network initialization, during which the mobile sink can construct its local view of the entire network efficiently. It is also essentially to build the global network topology in a distributed manner.

Phase 1: the sensor node closest to the mobile sink's initial position will be chosen as the center node. The center node broadcasts a beacon to its neighbors with its own physical location, inviting neighboring nodes to act as its child nodes. After a neighbor node receives such an invitation, it may face two different choices. If this node already has a parent, a message saying NO will be sent back to the center node. Otherwise, it sends out a messages saying YES with its own location and broadcasts a new beacon to search its own child nodes. Such a process advances at each sensor node side iteratively, until phase 2 begins.

Phase 2: If a sensor node does not receive any YES message, it will inform the size of its subtree to update the local view for the topology of its parent. Once a sensor node receives updating messages from all child nodes, it immediately updates the topology information stored locally and sends the updated result to its parent. Such a process continues until the center node completes the updating.

To better understand the network initialization process, we provide an example in Figure 4. Initially, S_0 is selected to be the center node because it is the closest one to the mobile sink. S_0 broadcasts a beacon denoted by B. In Figure 4(a), S_1 and S_2 receive such a message. Right now, they do not have their own parent nodes and thus return messages YES to S_0. Then S_1 broadcasts a new beacon. S_2, S_3 and S_4 will receive S_1's beacon. So far, S_2 already has a parent node. As a result, it responds a message No to S_1. On the other hand, since both S_3 and S_4 do not have their parent nodes yet, they send out messages YES. In Figure 4(b), S_2, S_3, and S_4 do not receive any YES message and they enter phase 2. In phase 2, S_2, S_3, and S_4 inform their parent nodes the number of sensor nodes in their own routing subtrees. Once S_1 successfully obtains such information from all its child nodes (i.e., S_3 and S_4), it immediately updates its own local view of the network topology and sends the updated result to its parent node (i.e., S_0) as shown in Figure 4(c). The

two-phase initialization is completed when the center node S_0 gets responses from all its child nodes.

After the network initialization, the information stored at each sensor node includes its location, workload (i.e., the number of sources in all the subtrees rooted at its child nodes), parent node's location, and child node's location. We define the *workload* of a sensor node as the total number of nodes in the local routing tree rooted at this node itself. We take node S_1 for example, where S_1 = { *location*: (x_1,y_1); *workload*: 3; *parent node*: $S_0(x_0,y_0)$; *child nodes set*: { $S_3(x_3,y_3)$, $S_4(x_4,y_4)$ } }.

B. Optimizing the Rendezvous Point Placement

To determine the location of each RP in RPS-LB-D, the mobile sink iteratively eliminates the workload bottleneck via allowing the current busiest RP to invite another nearby node as a new RP to share its burden until the trajectory length reaches its maximum moving distance *L*. Essentially, the strategy breaking the workload bottleneck in RPS-LB-D is rather similar to the strategy growing the median subtree towards the largest branch in RPS-LB.

In RPS-LB-D, the mobile sink only knows the information of location and workload of RPs. For instance, in Figure 4(a), the mobile sink initially selects the center node S_0 to be the first RP. In such a case, the current RP's information obtained by the mobile sink is: *RP-list* = { RP_1:(*location*: $S_0(0,0)$; *workload*: 5)}. Then the mobile sink moves to the RP with the heaviest word load and asks it to invite one neighboring node within the range of *d* to serve as a new RP.[3] The neighbor with the largest workload will be selected as the new RP and its own information (including its location and workload) will be immediately sent back to the mobile sink. In Figure 4, the busiest RP is S_0 and it invites S_1, who has the largest load 3 among all its child nodes, as a new RP. After becoming a new RP, S_1 can collect data from S_3 and S_4, which largely mitigates the original heavy workload of S_0. Updates on the routing tree should also be done at this time, i.e., now S_1 does not have a parent node and its received data will thus not be relayed. S_0 deletes S_1 from its child nodes list and does not wait for the data from S_1.

At the beginning, the mobile sink invites the nearest node (center) to become the first RP. It then executes the RPS-LB-D algorithm to recruit the new RPs. The mobile sink keeps moving towards the RP with the heaviest workload until the length of the trajectory reaches the maximum moving distance *L*. After reaching this RP, the mobile sink asks it to invite one nearby node to be a new RP. Once the new RP is selected, some necessary updates should be performed at both the sensor and the mobile sink sides. After the trajectory is determined, the mobile sink traverses along the formed moving trajectory and collects data at each by visiting all selected RPs. Sensory data are sent along the child-to-parent path unless reach the corresponding RP.

[3] The range *d* can be computed by $d = (L - TSP(R))/2$, where *R* is the set of RPs retrieved from *RP-list*, and *TSP(R)* is a TSP solver returning the length of the TSP tour that visits all nodes in *R*. Clearly, *d* is a key parameter guaranteeing the newly included RP and original RPs can all be visited by the new trajectory tour no longer than *L*. Its correctness can be proven by **Theorem 3**.

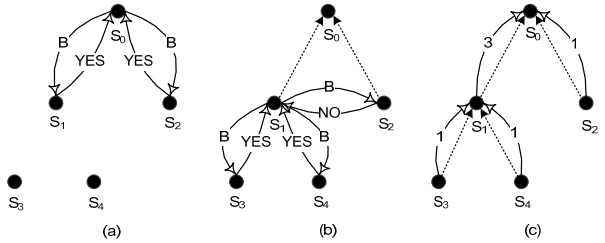

Figure 4 An example of network initialization in RPS-LB-D

VI. Performance Evaluation

A. Simulation Settings

In this section, we evaluate the performance of RPS-LB and RPS-LB-D by comparing them with two well-known mobility-assisted approaches named the Nearest-Neighbor based heuristic (NN) [16] and the Rendezvous Planning with Utility-based Greedy heuristic (RP-UG) [17]. Without loss of generality, the RPS-LB-D algorithm starts from a random initial position (i.e., the center node) to determine the location of sequent RPs. In NN, the mobile sink always travels to the nearest source that is closest to the current source that has been visited. The sources that are not visited by the sink connect to the closest source on the sink tour. As there is no any RP in NN, it serves as the benchmark clarifying the effectiveness of exploiting sink mobility in data collection. RP-UG is a recently proposed centralized network protocol exploiting rendezvous nodes and mobile elements to improve the efficiency of gathering data in WSNs. This approach attempts to minimize the total energy consumption of data delivery under the assumption that the cost of sending a packet is proportional to its traversal distance. As such energy model is also adopted by RPS-LB and RPS-LB-D, RP-UG can therefore serve as a suitable benchmark to evaluate our proposals in comparison with the state-of-the-art related work. In RP-UG, rendezvous nodes are determined in an iterative manner. In each iteration, RP-UG expands the visiting tour for mobile element to include more source(s) with the largest utility serving as the new rendezvous node(s). The utility is defined as the ratio of amount of saving energy to the extended length of tour. As the mobile element tour must contain a fixed base station for uploading data, we place the station at a random position in our network.

In simulations, varied numbers of sensors are densely distributed in a $500m \times 500m$ target region to guarantee the connectivity of network. After the initialization, an MST shaped global routing tree has been constructed to connect all sources. A source generates one data sample within a data collection period and needs to send all accumulated (including received) samples to its corresponding RP. We set the radio transmission radius of a sensor as $100m$. As the energy consumption of the wireless transmission is proportional to its Euclidean distance between the pair of source and destination nodes, we analyze the energy efficiency performance of various algorithms by comparing their total distances of data delivery paths. The *network lifetime* is quantified by the time duration from the network starts to work until the first node depletes its energy. Since energy is mainly consumed by wireless transmission, energy cost at each sensor side is

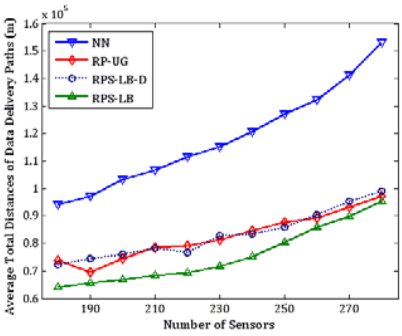

Figure 5 Total distances of data delivery paths vs. Number of source nodes.

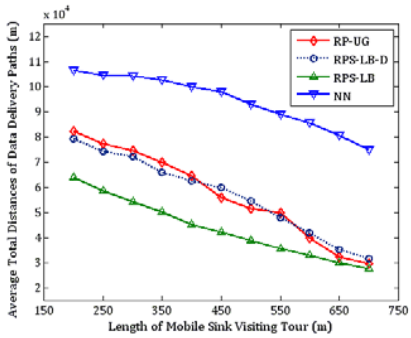

Figure 6 Total distances of data routing paths vs. Length of mobile sink visiting tour.

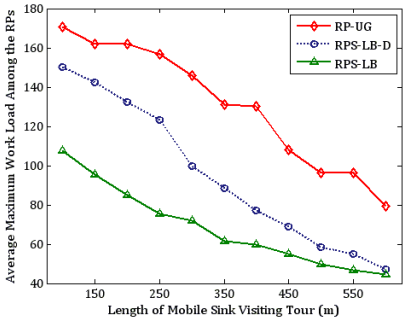

Figure 7 Average maximum workloads among the RPs vs. Length of mobile sink visiting tour.

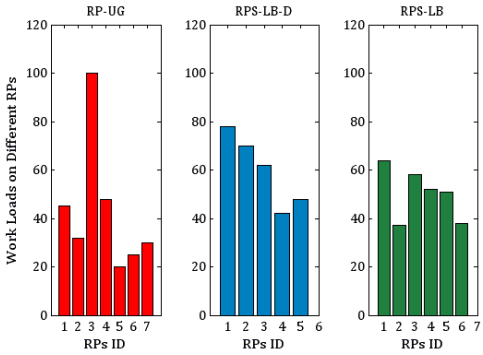

Figure 8 Workloads on the RPs in different algorithms.

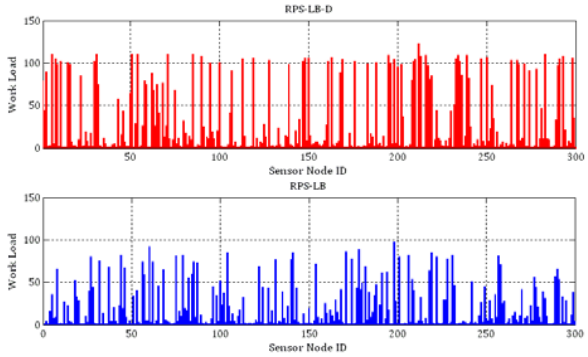

Figure 9 Workloads on sensor nodes in RPS-LB-D and RPS-LB.

proportional to its workload to relay sensory data. We define the *workload* of a sensor node as the number of sources in the local routing trees associated to this node. Hence, the lifetime of a network is reversely proportional to the maximum workload among all the sensor nodes. All the evaluation results are averaged based on 10 different runs.

B. The Performance of RPS-LB and RPS-LB-D

Figure 5 compares the total distances of data delivery paths formed by different algorithms with varied size of networks. The size here is defined as the number of sources deployed. In this experiment, the length of mobile sink visiting tour L is set to be 300m. According to Figure 5, it is clear to see that all rendezvous-based approaches achieve significant better performances than the one that just exploits sink mobility but not rendezvous nodes. In addition, the gap from NN to RP-UG or RPS-LB or RPS-LB-D is expanded as more source sensors are involved. When the network is growing larger, the energy efficiencies achieved by RPS-LB and RPS-LB-D are both boosting. It indicates that RPS-LB-D and RPS-LB can effectively reduce the energy consumption by taking advantage of the advanced reference structure, i.e. median subtree, proposed in this paper. Compared with the centralized algorithm RPS-LB and RP-UG, the distributed version RPS-LB-D also works well (e.g., its performance is even better than RP-UG in some cases) with only a slight performance distortion in terms of the delivery path length summation. It is because there is not an essential difference between RPS-LB and RPS-LB-D in protocol design except the root of median subtree, i.e., the root is the median in RPS-LB but a random node in RPS-LB-D. Moreover, we also find that RPS-LB outperforms RPS-LB-D and RP-UG especially in the case that the network is not so big. However, their performance gaps are gradually narrowed with the increase of network size. Such phenomenon is consistent with the implication of the approximation ratio of RPS-LB driven in Section IV.D.

We then evaluate the performances achieved by different algorithms with different data delivery deadlines. As we have mentioned above, such delay bound is equal to the sink traversal time, and can be finally mapped to the length of sink tour given the sink movement speed in average. Figure 6 illustrates the relationship between the total distances of data delivery paths and the length of sink tour. 200 source nodes are randomly deployed. All algorithms' performance becomes better when the sink mobility is enhanced. Consistent with the result in Figure 5, RPS-LB is superior to other three competitors. Although RPS-LB-D is a distributed algorithm, its performance is very close to the one gained by RP-UG. This is another indication of the effectiveness of RPS-LB and RPS-LB-D to reduce the energy consumption, which implies that the network can enjoy a longer lifetime in RPS-LB or RPS-LB-D compared with other two algorithms. Figure 7 shows the relationship between the average maximum workloads among RPs and the length of sink tour. Similar to Figure 6, 200 sources are randomly deployed in the field as well. According to Figure 7, the average maximum workload incurred by three rendezvous-based algorithms all decrease with the increase of L. More specifically, when the tour is short, the load is extremely heavy in RP-UG and RPS-LB-D, i.e., their workload all exceed 160. As load balancing is not considered in RP-UG, the

bottleneck node could not be eliminated even when a source is recruited to be a new rendezvous node. However, as L increases, RPS-LB-D rapidly cuts down its maximum workload and largely narrows the gap with RPS-LB. This implies that the distributed load balancing strategy in RPS-LB-D can effectively mitigate the workload on the bottleneck RP.

To further verify the effectiveness of our proposals, we show the snapshot of the workload on each RP in three different protocols. In this experiment, every node has been assigned a unique ID number beforehand. 300 source nodes are randomly deployed and the tour length is set to be 600m. In Figure 8, the variance of the workload on different RPs is very large in RP-UG, which is not as smooth as RPS-LB-D or RPS-LB. Our proposals are mainly benefited from the workload balancing consideration in trajectory planning.

In Figure 9, we take a fine look at the difference between RPS-LB-D and RPS-LB. Figure 9 compares the workload of each sensor node in RPS-LB-D with that in RPS-LB. According to the result, the workload of sensor nodes in RPS-LB is more uniform than that in RPS-LB-D. The reason for this phenomenon is that RPS-LB-D expands the trajectory from the node with the shortest distance to the mobile sink's initial location while RPS-LB expands the trajectory from the median. In addition, Figure 9 shows that the workload of sensor nodes is well balanced in both RPS-LB and RPS-LB-D as expected by the optimization objective in our original problem definition.

VII. CONCLUSION

In this paper, we study the data collection problem for the mobile sink in wireless sensor networks. We formulate such a problem as a joint optimization problem of both mobile sink trajectory planning and network load balancing. We prove that such a problem is NP-hard and propose an approximation algorithm RPS-LB to approach the optimal solution. We prove that RPS-LB satisfies energy saving and sustainable design requirements. Moreover, the derived approximation ratio validates the performance of RPS-LB. To further enhance the applicability of the proposed algorithm, we relax the assumption of the location information of each sensor node is obtained by the mobile sink and propose a localized, fully distributed version RPS-LB-D. Compared with existing works, the proposed RPS-LB guarantees low total energy consumption over the network and achieves much more balanced overhead across different RPs. In the future, we will implement and evaluate our algorithms on real testbeds.

ACKNOWLEDGEMENT

This work is supported by the NSFC under grants No. 60803152 and No.61100075, the key research project of Ministry of Education grant No.109144, the fundamental research fund for the Central Universities No.72104238 and No.K50510230004, COE SUG/RSS 20Aug2010 13/14 in Nanyang Technological University of Singapore, and key technologies of electromagnetic spectrum monitoring based on wireless sensor networks grant No. 2010ZX03006-002-04.

REFERENCES

[1] Yanmin Zhu and Lionel M. Ni, "Probabilistic Approach to Provisioning Guaranteed QoS for Distributed Event Detection", In *Proceedings of IEEE INFOCOM*, 2008.

[2] Zheng Yang, and Yunhao Liu, "Quality of Trilateration: Confidence based Iterative Localization", IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. 21, No. 5, May 2010,Pages 631-640.

[3] Kebin Liu, Qiang Ma, Xibin Zhao, Yunhao Liu, "Self-Diagnosis for Large Scale Wireless Sensor Networks", In *IEEE INFOCOM*, 2011.

[4] Wei Dong, Yunhao Liu, Xiaofan Wu, Lin Gu, and Chun Chen, "Elon: Enabling Efficient and Long-Term Reprogramming in Wireless Sensor Networks", In *ACM SIGMETRICS*, 2010.

[5] Jie Lian, Yunhao Liu, K. Naik, and Lei Chen, "Virtual Surrounding Face Geocasting with Guaranteed Message Delivery for Ad Hoc and Sensor Networks", IEEE/ACM Transactions on Networking (TON), Vol. 17, No. 1, February 2009, Pages 200-211.

[6] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G.S.Sukhatme, "Robomote: enabling mobility in sensor networks," In *Proceedings of IEEE IPSN*, 2005.

[7] D.J. Chang and E.K. Morlok. "Vehicle speed profiles to minimize work and fuel consumption," Journal of Transportation Engineering, 131(3), 2005.

[8] X. Li, A. Nayak, I. Stojmenovic, "Exploiting Actuator Mobility for Energy efficient Data Collection in Delay-Tolerant Wireless Sensor Networks," In *5th International Conference on Networking and Services*, 2009.

[9] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using Predictable Observer Mobility for Power Efficient Design ofSensor Networks," In *Proceedings of IEEE IPSN*, 2003.

[10] Y. Gu, D. Bozdag; R. W. Brewer, and E. Ekici, "Dataharvesting with mobile elements in wireless sensor networks," Computer Networks, vol. 50, no. 17, 2006.

[11] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling with dynamic deadlines," IEEE Transactions on Mobile Computing, vol. 6, no. 4, 2007.

[12] W. Wang, V. Srinivasan, and K. C. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks," in *Proceedings of ACM MobiCom*, 2005.

[13] Y. Shi and Y.T. Hou, "Theoretical Results on Base Station Movement Problem for Sensor Networks,"In*Proceedings of IEEE INFOCOM*, 2008.

[14] A. Kansal, D.D. Jea, D. Estrin, and M.B. Srivastava, "Controllably mobile infrastructure for low energy embedded networks," IEEE Transactions on Mobile Computing, vol. 5, no. 8, 2006.

[15] J. Rao and S. Biswas, "Joint Routing and Navigation Protocols for Data Harvesting in Sensor Networks," In *Proceedings of IEEE MASS*, 2008.

[16] G. Xing, T. Wang, W. Jia, and M. Li. "Rendezvous Design Algorithms for Wireless Sensor Networks with a Mobile Base Station," In *Proceedings of ACM MobiHoc*, 2008.

[17] G. Xing, T. Wang, Z. Xie, and W. Jia "Rendezvous Planning in Mobility-assisted Wireless Sensor Networks," In *IEEE RTSS*, 2007.

[18] K. Almi'ani, A. Viglas, and L. Libman, "Energy-Efficient Data Gathering with Tour Length-Constrained Mobile Elements in Wireless Sensor Networks," In *Proceedings of IEEE Conference on Local Computer Networks (LCN)*, 2010.

[19] A.K. Kumar and K.M. Sivalingam, "Energy-Efficient Mobile Data Collection in WirelessSensor Networks with Delay Reduction usingWireless Communication," In *Proceedings of COMSNETS*, 2010.

[20] S. Gao, H. Zhang, and S.K. Das, "Efficient Data Collection in Wireless Sensor Networks with Path-Constrained Mobile Sinks," In *Proceedings of IEEE WoWMoM*, 2009.

[21] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Ultra-low power data storage for sensor networks," In *Proceedings of IEEE IPSN*,2006.

[22] S. Peng and W. Lo, "Efficient algorithms for finding a core of a tree with a specific length," Journal of Algorithms, 20:455-458, 1996.

[23] T. He, J. A. Stankovic, C. Lu, and T.F. Abdelzaher. "A spatio temporal communication protocol for wireless sensor networks," IEEE Transactions on Parallel and Distributed Systems, 16(10), 2005.

[24] Long-fei Shangguan, Luo Mai, Junzhao Du, Hui Liu, Wen He, "Energy-efficient Heterogeneous Data Collection in Mobile Wireless Sensor Networks", In *the PEMCT workshop adjunct with the proceedings of ICCCN*, 2011.