Mobile Data Gathering with Load Balanced Clustering and Dual Data Uploading in Wireless Sensor Networks

Miao Zhao, *Member, IEEE*, Yuanyuan Yang, *Fellow, IEEE*, and Cong Wang

Abstract—In this paper, a three-layer framework is proposed for mobile data collection in wireless sensor networks, which includes the sensor layer, cluster head layer, and mobile collector (called SenCar) layer. The framework employs distributed load balanced clustering and dual data uploading, which is referred to as LBC-DDU. The objective is to achieve good scalability, long network lifetime and low data collection latency. At the sensor layer, a distributed load balanced clustering (LBC) algorithm is proposed for sensors to self-organize themselves into clusters. In contrast to existing clustering methods, our scheme generates multiple cluster heads in each cluster to balance the work load and facilitate dual data uploading. At the cluster head layer, the inter-cluster transmission range is carefully chosen to guarantee the connectivity among the clusters. Multiple cluster heads within a cluster cooperate with each other to perform energy-saving inter-cluster communications. Through inter-cluster transmissions, cluster head information is forwarded to SenCar for its moving trajectory planning. At the mobile collector layer, SenCar is equipped with two antennas, which enables two cluster heads to simultaneously upload data to SenCar in each time by utilizing multi-user multiple-input and multiple-output (MU-MIMO) technique. The trajectory planning for SenCar is optimized to fully utilize dual data uploading capability by properly selecting polling points in each cluster. By visiting each selected polling point, SenCar can efficiently gather data from cluster heads and transport the data to the static data sink. Extensive simulations are conducted to evaluate the effectiveness of the proposed LBC-DDU scheme. The results show that when each cluster has at most two cluster heads, LBC-DDU achieves over 50 percent energy saving per node and 60 percent energy saving on cluster heads comparing with data collection through multi-hop relay to the static data sink, and 20 percent shorter data collection time compared to traditional mobile data gathering.

Index Terms—Wireless sensor networks (WSNs), data collection, load balanced clustering, dual data uploading, multi-user multiple-input and multiple-output (MU-MIMO), mobility control, polling point

1 Introduction

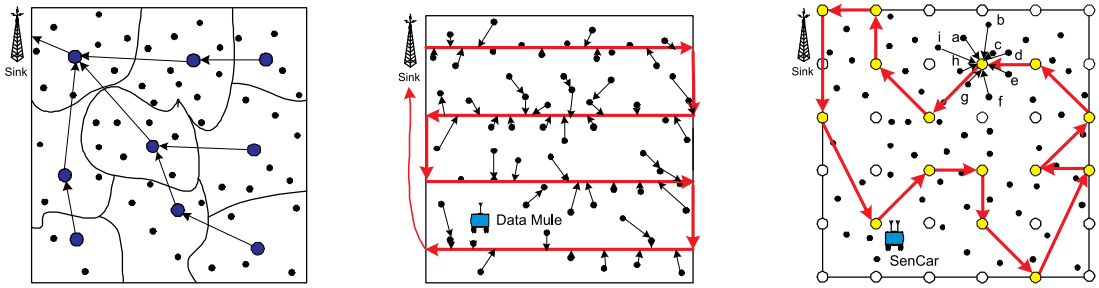
The proliferation of the implementation for low-cost, low-power, multifunctional sensors has made wireless sensor networks (WSNs) a prominent data collection paradigm for extracting local measures of interests [1], [2]. In such applications, sensors are generally densely deployed and randomly scattered over a sensing field and left unattended after being deployed, which makes it difficult to recharge or replace their batteries. After sensors form into autonomous organizations, those sensors near the data sink typically deplete their batteries much faster than others due to more relaying traffic. When sensors around the data sink deplete their energy, network connectivity and coverage may not be guaranteed. Due to these constraints, it is crucial to design an energy-efficient data collection scheme that consumes energy uniformly across the sensing field to achieve long network lifetime [3]. Furthermore, as sensing data in some applications are time-sensitive, data collection may be required to be performed within a specified time frame. Therefore, an efficient, large-scale data collection scheme should aim at good scalability, long network lifetime and low data latency.

Several approaches have been proposed for efficient data collection in the literature, see, for example, [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]. Based on the focus of these works, we can roughly divide them into three categories. The first category is the enhanced relay routing [4], [5], [6], [7], [8], [9], in which data are relayed among sensors. Besides relaying, some other factors, such as load balance, schedule pattern and data redundancy, are also considered. The second category organizes sensors into clusters and allows cluster heads to take the responsibility for forwarding data to the data sink (as shown in Fig. 1a) [10], [11], [12], [13], [14], [15], [16], [17], [18]. Clustering is particularly useful for applications with scalability requirement and is very effective in local data aggregation since it can reduce the collisions and balance load among sensors. The third category is to make use of mobile collectors to take the burden of data routing from sensors (as shown in Fig. 1b) [19], [20], [21], [22], [23], [24], [25].

Although these works provide effective solutions to data collection in WSNs, their inefficiencies have been noticed. Specifically, in relay routing schemes, minimizing energy consumption on the forwarding path does not necessarily prolong network lifetime, since some critical sensors on the path may run out of energy faster than others. In cluster-based schemes, cluster heads will inevitably consume

• *The authors are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, 11794, NY. E-mail: mzhao.ny@gmail.com, yuanyuan.yang@stonybrook.edu, cong.wang@stonybrook.edu.*

Manuscript received 10 Aug. 2013; revised 16 June 2014; accepted 25 June 2014. Date of publication 14 July 2014; date of current version 2 Mar. 2015. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2014.2338315



(a) Sensor clustering: achieving scalability and facilitating in-network aggregation and processing.

(b) Data Mule: taking over the routing burden from sensors to reduce the energy cost at sensor side.

(c) SenCar: optimizing on data collection tour for mobile collectors and minimizing relays at the sensors.

Fig. 1. Some examples of efficient data collection mechanisms in wireless sensor networks.

much more energy than other sensors due to handling intra-cluster aggregation and inter-cluster data forwarding. Though using mobile collectors may alleviate non-uniform energy consumption, it may result in unsatisfactory data collection latency. Based on these observations, in this paper, we propose a three-layer mobile data collection framework, named *Load Balanced Clustering and Dual Data Uploading (LBC-DDU)*. The main motivation is to utilize distributed clustering for scalability, to employ mobility for energy saving and uniform energy consumption, and to exploit Multi-User Multiple-Input and Multiple-Output (MU-MIMO) technique for concurrent data uploading to shorten latency.

The main contributions of this work can be summarized as follows. First, we propose a distributed algorithm to organize sensors into clusters, where each cluster has multiple cluster heads. In contrast to clustering techniques proposed in previous works [10], [11], [12], [13], our algorithm balances the load of intra-cluster aggregation and enables dual data uploading between multiple cluster heads and the mobile collector. Second, multiple cluster heads within a cluster can collaborate with each other to perform energy-efficient inter-cluster transmissions. Different from other hierarchical schemes [17], [18], in our algorithm, cluster heads do not relay data packets from other clusters, which effectively alleviates the burden of each cluster head. Instead, forwarding paths among clusters are only used to route small-sized identification (ID) information of cluster heads to the mobile collector for optimizing the data collection tour. Third, we deploy a mobile collector with two antennas (called *SenCar* in this paper) to allow concurrent uploading from two cluster heads by using MU-MIMO communication. The SenCar collects data from the cluster heads by visiting each cluster. It chooses the stop locations inside each cluster and determines the sequence to visit them, such that data collection can be done in minimum time. Our work mainly distinguishes from other mobile collection schemes [20], [21] in the utilization of MU-MIMO technique, which enables dual data uploading to shorten data transmission latency. We coordinate the mobility of SenCar to fully enjoy the benefits of dual data uploading, which ultimately leads to a data collection tour with both short moving trajectory and short data uploading time.

The rest of the paper is organized as follows. Section 2 performs a literature review of previous works. Section 3 presents the system overview of the proposed framework. Section 4 describes the distributed load balanced clustering algorithm on sensor nodes and Section 5 considers the cluster head layer. Section 6 focuses on data collection tour planning of SenCar. Finally, Section 7 provides performance evaluation results and Section 8 concludes the paper.

2 Related Works

2.1 Relay Routing and Clustering Schemes

Relay routing is a simple and effective approach to routing messages to the data sink in a multi-hop fashion. Cheng et al. [4] devised a coordinated transfer schedule by choosing alternate routes to avoid congestions. Wu et al. [8] studied the construction of a maximum-lifetime data gathering tree by designing an algorithm that starts from an arbitrary tree and iteratively reduces the load on bottleneck nodes. Xu et al. [5] studied deployments of relay nodes to elongate network lifetime. Gnewali et al. evaluated collection tree protocol (CTP) via testbeds in [6]. CTP computes wireless routes adaptive to wireless link status and satisfies reliability, robustness, efficiency and hardware independence requirements. However, when some nodes on the critical paths are subject to energy depletion, data collection performance will be deteriorated.

Another approach is to allow nodes to form into clusters to reduce the number of relays [10], [11], [12] , [13]. Heinzelman et al. [10] proposed a cluster formation scheme, named LEACH, which results in the smallest expected number of clusters. However, it does not guarantee good cluster head distribution and assumes uniform energy consumption for cluster heads. Younis and Fahmy [11] further proposed "HEED," in which a combination of residual energy and cost is considered as the metric in cluster head selection. HEED can produce well-distributed cluster heads and compact clusters. Gong et al. [12] considered energy-efficient clustering in lossy wireless sensor networks based on link quality. Amis et al. [13] addressed d-hop clustering with each node being at most d hops away from a cluster head. In these cluster-based schemes, besides serving as the aggregation point for local data collection, a cluster head also acts as a scheduler or controller for in-network processing. Zhang et al. [15] considered efficient scheduling of cluster heads to alleviate the collisions among different transmissions. Gedik et al. [17] and Liu et al. [18] explored the correlation of sensing data and dynamically partitioned

the sensor nodes into clusters. The cluster heads utilize the spatio-temporal correlation to minimize the readings for energy saving. Nevertheless, traditional single-head clustering schemes may not be compatible with MU-MIMO. Thus, for generality, we propose a load-balanced multi-head clustering algorithm in this paper.

2.2 Mobile Data Collections

Compared with data collection via a static sink, introducing mobility for data collection enjoys the benefits of balancing energy consumptions in the network and connecting disconnected regions. Shah et al. [19] investigated mobility under random walk where the mobile collector picks up data from nearby sensors, buffers and finally offloads data to the wired access point. However, random trajectory cannot guarantee latency bounds which is required in many applications. In [20], Jea et al. further proposed to control data mules to traverse the sensing field along parallel straight lines and collect data from nearby sensors with multi-hop transmissions as shown in Fig. 1b. This scheme works well in a uniformly distributed sensor network. To achieve more flexible data gathering tour for mobile collectors, Ma and Yang [16] proposed an efficient moving path planning algorithm by determining some turning points on the straight lines, which is adaptive to the sensor distribution and can effectively avoid obstacles on the path. In [21], they alternatively proposed a single-hop data gathering scheme to pursue the perfect uniformity of energy consumption among sensors (see Fig. 1c), where a mobile collector called SenCar is optimized to stop at some locations to gather data from sensors in the proximity via single-hop transmission. The work was further extended in [24] to optimize the data gathering tour by exploring the tradeoff between the shortest moving tour of SenCar and the full utilization of concurrent data uploading among sensors. Furthermore, Somasundara et al. proposed an algorithm [25] to study the scheduling of mobile elements such that there is no data loss due to buffer overflow. Although these works consider utilizing mobile collectors, latency may be increased due to data transmission and mobile collector's traveling time. Thus, in this paper, we exploit MU-MIMO to reduce data transmission time for mobile data collection.

2.3 MU-MIMO in WSNs

The feasibility of employing MIMO techniques in wireless sensor networks is envisioned in [27], [28], [29]. Due to difficulties to mount multiple antennas on a single sensor node, MIMO is adopted in WSNs to seek cooperations from multiple nodes to achieve diversity and reduce bit error rate. An overview of MIMO-based scheduling algorithms to coordinate transmissions was discussed in [26]. Another challenge in MIMO is that the energy consumption in circuits could be higher than a traditional Single-Input-Single-Output (SISO) approach. In [27], it was demonstrated that MIMO can outperform SISO when the transmission distance is larger than certain thresholds (e.g., 25 m). In [28], it was shown that with proper designs of system parameters, significant energy saving can be achieved with MIMO techniques. To compensate energy consumptions in the circuit, optimization of transmission time and modulation parameters was presented in [29]. In our framework, since it is not difficult to deploy two antennas on the mobile collector, when a compatible pair of transmitting nodes and the locations of the mobile collector are given, we can enable MU-MIMO uploading to the mobile collector to greatly reduce data collection latency.

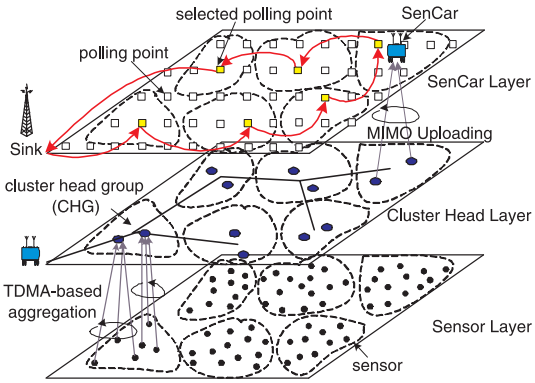


Fig. 2. Illustration of the LBC-DDU framework.

3 System Overview

An overview of LBC-DDU framework is depicted in Fig. 2, which consists of three layers: sensor layer, cluster head layer and SenCar layer.

The sensor layer is the bottom and basic layer. For generality, we do not make any assumptions on sensor distribution or node capability, such as location-awareness. Each sensor is assumed to be able to communicate only with its neighbors, i.e., the nodes within its transmission range. During initialization, sensors are self-organized into clusters. Each sensor decides to be either a *cluster head* or a *cluster member* in a distributed manner. In the end, sensors with higher residual energy would become cluster heads and each cluster has at most M cluster heads, where M is a system parameter. For convenience, the multiple cluster heads within a cluster are called a *cluster head group* (CHG), with each cluster head being the *peer* of others. The algorithm constructs clusters such that each sensor in a cluster is one-hop away from at least one cluster head. The benefit of such organization is that the intra-cluster aggregation is limited to a single hop. In the case that a sensor may be covered by multiple cluster heads in a CHG, it can be optionally affiliated with one cluster head for load balancing.

To avoid collisions during data aggregation, the CHG adopts time-division-multiple-access (TDMA) based technique to coordinate communications between sensor nodes. Right after the cluster heads are elected, the nodes synchronize their local clocks via beacon messages. For example, all the nodes in a CHG could adjust their local clocks based on that of the node with the highest residual energy. After local synchronization is done, an existing scheduling scheme [30], [31] can be adopted to gather data from cluster members. Note that only intra-cluster synchronization is needed here because data are collected via SenCar. In the case of imperfect synchronization, some hybrid techniques to combine TDMA with contention-based access protocols (Carrier Sense Multiple Access (CSMA)) that listen to the medium before transmitting are required. For example, hybrid

protocols like Z-MAC can be utilized [32] to enhance the strengths and offset the weaknesses of TDMA and CSMA. Upon the arrival of SenCar, each CHG uploads buffered data via MU-MIMO communications and synchronizes its local clocks with the global clock on SenCar via acknowledgement messages. Finally, periodical re-clustering is performed to rotate cluster heads among sensors with higher residual energy to avoid draining energy from cluster heads.

The cluster head layer consists of all the cluster heads. As aforementioned, inter-cluster forwarding is only used to send the CHG information of each cluster to SenCar, which contains an identification list of multiple cluster heads in a CHG. Such information must be sent before SenCar departs for its data collection tour. Upon receiving this information, SenCar utilizes it to determine where to stop within each cluster to collect data from its CHG. To guarantee the connectivity for inter-cluster communication, the cluster heads in a CHG can cooperatively send out duplicated information to achieve spatial diversity, which provides reliable transmissions and energy saving [27]. Moreover, cluster heads can also adjust their output power for a desirable transmission range to ensure a certain degree of connectivity among clusters.

The top layer is the SenCar layer, which mainly manages mobility of SenCar. There are two issues to be addressed at this layer. First, we need to determine the positions where SenCar would stop to communicate with cluster heads when it arrives at a cluster. In LBC-DDU, SenCar communicates with cluster heads via single-hop transmissions. It is equipped with two antennas while each sensor has a single antenna and is kept as simple as possible. The traffic pattern of data uploading in a cluster is many-to-one, where data from multiple cluster heads converge to SenCar. Equipped with two receiving antennas, each time SenCar makes dual data uploading whenever possible, in which two cluster heads can upload data simultaneously. By processing the received signals with filters based on *channel state information*, SenCar can successfully separate and decode the information from distinct cluster heads.

To collect data as fast as possible, SenCar should stop at positions inside a cluster that can achieve maximum capacity. In theory, since SenCar is mobile, it has the freedom to choose any preferred position. However, this is infeasible in practice, because it is very hard to estimate channel conditions for all possible positions. Thus, we only consider a finite set of locations. To mitigate the impact from dynamic channel conditions, SenCar measures channel state information before each data collection tour to select candidate locations for data collection. We call these possible locations SenCar can stop to perform concurrent data collections *polling points*. In fact, SenCar does not have to visit all the polling points. Instead, it calculates some polling points which are accessible and we call them *selected polling points*. In addition, we need to determine the sequence for SenCar to visit these selected polling points such that data collection latency is minimized. Since SenCar has pre-knowledge about the locations of polling points, it can find a good trajectory by seeking the shortest route that visits each selected polling point exactly once and then returns to the data sink.

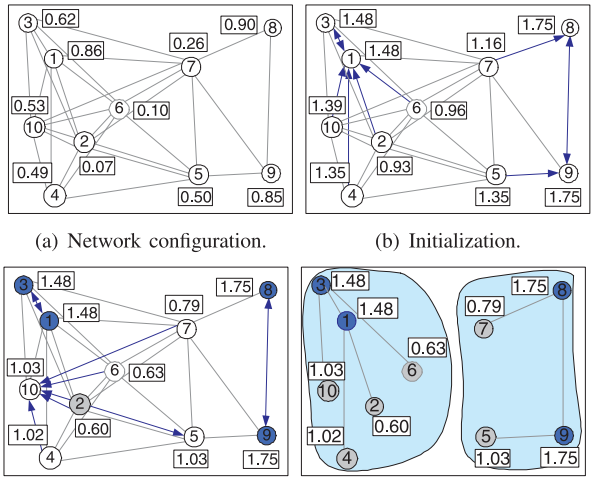
The proposed framework aims to achieve great energy saving and shortened data collection latency, which has the potential for different types of data services. Although traditional designs of WSNs can support low-rate data services, more and more sensing applications nowadays require high-definition pictures and audio/video recording, which has become an overwhelming trend for next generation sensor designs. For example, in the scenario of military defense, sensors deployed in reconnaissance missions need to transmit back high-definition images to identify hostile units. Delays in gathering sensed data may not only expose sensors or mobile collector to enemy surveillance but also depreciate the time value of gathered intelligence. Using MU-MIMO can greatly speed up data collection time and reduce the overall latency. Another application scenario emerges in disaster rescue. For example, to combat forest fire, sensor nodes are usually deployed densely to monitor the situation. These applications usually involve hundreds of readings in a short period (a large amount of data) and are risky for human being to manually collect sensed data. A mobile collector equipped with multiple antennas overcomes these difficulties by reducing data collection latency and reaching hazard regions not accessible by human being. Although employing mobility may elongate the moving time, data collection time would become dominant or at least comparable to moving time for many high-rate or densely deployed sensing applications. In addition, using the mobile data collector can successfully obtain data even from disconnected regions and guarantee that all of the generated data are collected.

4 Sensor Layer: Load Balanced Clustering

In this section, we present the distributed load balanced clustering algorithm at the sensor layer. The essential operation of clustering is the selection of cluster heads. To prolong network lifetime, we naturally expect the selected cluster heads are the ones with higher residual energy. Hence, we use the percentage of residual energy of each sensor as the initial clustering priority. Assume that a set of sensors, denoted by $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, are homogeneous and each of them independently makes the decision on its status based on local information. After running the LBC algorithm, each cluster will have at most M (≥ 1) cluster heads, which means that the size of CHG of each cluster is no more than M. Each sensor is covered by at least one cluster head inside a cluster. The LBC algorithm is comprised of four phases: (1) Initialization; (2) Status claim; (3) Cluster forming and (4) Cluster head synchronization. Next, we describe the operation through an example in Fig. 3, where a total of 10 sensors (plotted as numbered circles in Fig. 3a) are labeled with their initial priorities and the connectivity among them is shown by the links between neighboring nodes.

4.1 Initialization Phase

In the initialization phase, each sensor acquaints itself with all the neighbors in its proximity. If a sensor is an isolated node (i.e., no neighbor exists), it claims itself to be a cluster head and the cluster only contains itself. Otherwise, a sensor, say, s_i, first sets its status as "tentative" and its initial priority by the percentage of residual energy. Then, s_i sorts its neighbors by their initial priorities and picks $M-1$



(a) Network configuration. (b) Initialization.

(c) Status claim. (d) Cluster forming.

Fig. 3. An example of LBC algorithm with $M = 2$.

neighbors with the highest initial priorities, which are temporarily treated as its *candidate peers*. We denote the set of all the candidate peers of a sensor by $\mathcal{A}$. It implies that once s_i successfully claims to be a cluster head, its up-to-date candidate peers would also automatically become the cluster heads, and all of them form the CHG of their cluster. s_i sets its priority by summing up its initial priority with those of its candidate peers. In this way, a sensor can choose its favorable peers along with its status decision. Fig. 3b depicts the initialization phase of the example, where M is set to 2, which means that each sensor would pick one neighbor with the highest initial priority as its candidate peer. We use the out-going arrow to indicate the choice of each sensor. For instance, s_8 is chosen to be the peer of s_7 since it is the one with the highest initial priority among all the neighbors of s_7. Accordingly, s_7 sets its priority to the sum of the initial priorities of s_7 and s_8. The pseudo-code describing the initialization phase of a sensor is given in Algorithm 1, and notations used in the pseudo-codes are listed in Table 1 for reference.

Algorithm 1. Phase I: Initialization

1: My.$\mathcal{N}$ $\longleftarrow$ $\{v | v$ lies in my transmission range, $v \in \mathcal{S}\}$;
2: **if** $My.\mathcal{N} = \Phi$ **then**
3: Set My.cluster_head to My.id;
4: Set My.status to *cluster_head*;
5: **else**
6: My.init_prio $\longleftarrow$ E_{res}/E_{tot};
7: My.cluster_head $\longleftarrow$ 0;
8: My.status $\longleftarrow$ *tentative*;
9: My.$\mathcal{A}$ $\longleftarrow$ $\{v | v \in$ Can_Peers($\mathcal{N}$)$\}$;
10: My.prio $\longleftarrow$ My.init_prio+ $\sum_{v \in \text{My}.\mathcal{A}} v$.init_prio;
11: My.$\mathcal{B}$, My.$\mathcal{C}$ $\longleftarrow$ Φ;
12: Iter $\longleftarrow$ 0;

4.2 Status Claim

In the second phase, each sensor determines its status by iteratively updating its local information, refraining from promptly claiming to be a cluster head. We use the node degree to control the maximum number of iterations for each sensor. Whether a sensor can finally become a cluster head primarily depends on its priority. Specifically, we partition the priority into three zones by two thresholds, τ_h and τ_m ($\tau_h > \tau_m$) , which enable a sensor to declare itself to be a cluster head or member, respectively, before reaching its maximum number of iterations. During the iterations, in some cases, if the priority of a sensor is greater than τ_h or less than τ_m compared with its neighbors, it can immediately decide its final status and quit from the iteration.

We denote the potential cluster heads in the neighborhood of a sensor by a set $\mathcal{B}$. In each iteration, a senor, say, s_i, first tries to probabilistically include itself into $s_i.\mathcal{B}$ as a *tentative cluster head* if it is not in already (Algorithm 2, lines 2-5). Once successful, a packet includes its node ID and priority will be sent out and the sensors in the proximity will add s_i as their potential cluster heads upon receiving the packet. Then, s_i checks its current potential cluster heads. If they do exist, there are two cases for s_i to make the final status decision, otherwise, s_i would stay in the tentative status for the next round of iteration.

The first case (Algorithm 2, lines 8-11) is that s_i has reached its maximum number of iterations and it prevails over others in $s_i.\mathcal{B}$ with the highest priority. Then s_i will claim to be a cluster head in this case. We call this process *self-driven status transition*. Also, s_i will announce its current candidate peers to be cluster heads by broadcasting a packet including an ID list, which is referred to as *the peer-driven status transition*. Once a sensor in the neighborhood receives a status packet, it may need to update its status to the cluster head. The detailed description of handling received packets in each sensor can be found in Algorithm 4, *Function recv_pkt*.

TABLE 1
Notations Used in Section 4

Notation	Meaning
$\mathcal{S}$	Set of sensors, where $\mathcal{S}[i]$ represents sensor s_i;
My.$\mathcal{N}$	Set of neighbors in my proximity;
My.$\mathcal{A}$	Set of my candidate cluster head peers;
My.$\mathcal{B}$	Set of my potential cluster heads;
My.$\mathcal{C}$	Set of my cluster members;
My.init_prio	My initial priority;
My.prio	My current priority;
My.status	My current status (i.e., cluster_head, member, or tentative);
M	Maximum number of heads in a cluster;
E_{res}, E_{tot}	Current residual and maximum energy of a sensor;
τ_h, τ_m	Priority threshold for claiming to be a cluster head, member respectively;
Can_Peers()	Function of finding $M-1$ elements with the highest priorities among the input set as candidate peers;
Highest_Prio()	Function of finding the element with highest priority among the input set;
Lowest_Prio()	Function of finding the element with lowest priority among the input set;
Fnl_N()	Function of finding the subset of the input set, in which all the elements are cluster heads;
Rand_one()	Function of randomly choosing an element from the input set.

Algorithm 2. Phase II: Status claim

1: **while** $My.\mathcal{N}| > 0 \& Iter \leq |My.\mathcal{N}| \& My.status = tentative$ **do**
2: **if** $My.prio > \sum_{i=1}^{M} Rand(1) \& My \notin My.\mathcal{B}$ **then**
3: Add myself to My.$\mathcal{B}$;
4: send_pkt (1, My.id, null, *tentative*, My.prio);
5: /*send_pkt(msg.type,node id,node list,node status, node priority);*/
6: **if** $My.\mathcal{B} \neq \Phi$ **then**
7: **if** $Highest\_Prio(My.\mathcal{B}) = My.id$ **then**
8: **if** $Iter = |My.\mathcal{N}|$ **then**
9: My.status $\longleftarrow$ *cluster_head*;
10: recv_pkt ();
11: send_pkt (2, My.id, ID_List(My.$\mathcal{A}$), *cluster_head*, My.prio);
12: **else if** $Lowest\_Prio(My.\mathcal{B}) = My.id$ & $Fnl\_N(My.\mathcal{B}) \neq \Phi$ **then**
13: **if** $My.prio \leq \tau_m$ **then**
14: My.status $\longleftarrow$ *cluster_member*;
15: **else if** $My.prio > \tau_h$ **then**
16: My.status $\longleftarrow$ *cluster_head*;
17: recv_pkt ();
18: send_pkt (2, My.id, ID_List(My.$\mathcal{A}$), τ_h, My.prio);
19: Iter $\longleftarrow$ Iter + 1;

The second case (Algorithm 2, lines 11-12) is that s_i is the one with the lowest priority and there exist some cluster heads in $s_i.\mathcal{B}$. In this case, if the priority of s_i is less than or equal to τ_m, it is clearly not qualified to be a cluster head. Hence, it quits the iteration and claims to be a cluster member. It is safe for such "retirement," since there are already some cluster heads in its neighborhood and it can optionally affiliate with one of them at a later time. Moreover, when s_i does not have any potential candidate cluster head in the current iteration and its priority is already high enough (above τ_h), it can immediately claim to be a cluster head (Algorithm 2, lines 15-18). Fig. 3c shows the result of phase II for the example with τ_h and τ_m set to 1.8 and 0.6, respectively. (s_1, s_3) and (s_8, s_9) become the cluster heads while s_2 is a cluster member with the lowest priority. Different from the initialization phase, it is also shown that the candidate peers of each sensor have been updated. For example, sensor s_5 is still tentative at the end of phase II, which initially considered s_9 as its candidate peer, but later switched to s_{10} as its new peer when s_9 reached its final status.

4.3 Cluster Forming

The third phase is cluster forming that decides which cluster head a sensor should be associated with. The criteria can be described as follows: for a sensor with tentative status or being a cluster member, it would randomly affiliate itself with a cluster head among its candidate peers for load balance purpose. In the rare case that there is no cluster head among the candidate peers of a sensor with tentative status, the sensor would claim itself and its current candidate peers as the cluster heads. The details are given in Algorithm 3. Fig. 3 d shows the final result of clusters, where each cluster has two cluster heads and sensors are affiliated with different cluster heads in the two clusters.

Algorithm 3. Phase III: Cluster forming

1: **if** *My.status* = *cluster_head* **then**
My.cluster_head$\longleftarrow$My.id;
2: **else**
3: recv_pkt ();
4: My.$\mathcal{B}$ $\longleftarrow$Fnl_N(My.$\mathcal{B}$);
5: **if** $My.\mathcal{B} \neq \Phi$ **then**
6: My.status $\longleftarrow$ *cluster_member*;
7: My.cluster_head$\longleftarrow$Rand_one(My.$\mathcal{B}$).id;
8: send_pkt (3, My.id, My.cluster_head, *cluster_member*, My.init_prio);
9: **else**
10: My.status $\longleftarrow$ *cluster_head*;
11: My.cluster_head $\longleftarrow$ My.id;
12: send_pkt (2, My.id, ID_List(My.$\mathcal{A}$), *cluster_head*, My.prio);

Algorithm 4. recv_pkt

1: **for** *each recvd PKT with My.id* $\neq$ *PKT.src_id* **do**
2: **if** *PKT.type* = 1 **then**
3: Add sensor $\mathcal{S}$[PKT.src_id] to My.$\mathcal{B}$;
4: **else if** *PKT.type* = 2 **then**
5: Add sensor $\mathcal{S}$[PKT.src_id] to My.$\mathcal{B}$;
6: **if** $\mathcal{S}$[*PKT.src_id*]$\in$ *My.*$\mathcal{A}$
7: Remove $\mathcal{S}$[PKT.src_id] from My.$\mathcal{A}$;
8: Find a sensor u from My.$\mathcal{N}$, which is not in current My.$\mathcal{A}$ and its status is tentative with the highest initial priority;
9: **if** u *exists* **then** My.$\mathcal{A}$ $\longleftarrow$ My.$\mathcal{A} \bigcup \{u\}$;
10: **if** $i = 1$ *to* $M - 1$ **do**
11: **if** *My.id* = *PKT.src_peerlist*[i] **then**
12: My.status $\longleftarrow$ *cluster_head*;
13: My.prio $\longleftarrow$PKT.src_prio;
14: My.$\mathcal{A}$ $\longleftarrow$ $\mathcal{S}$[PKT.src_id].$\mathcal{A}$;
15: send_pkt(1, My.id, *cluster_head*, My.prio);
16: **else if** $\mathcal{S}$[*PKT.src_peerlist[i]*] $\in$ *My.*$\mathcal{N}$ **then**
17: Add $\mathcal{S}$[PKT.src_peerlist[i]] to My.$\mathcal{B}$;
18: **if** $\mathcal{S}$[*PKT.src_peerlist*[i]] $\in$ *My.*$\mathcal{A}$ **then**
19: Remove $\mathcal{S}$[PKT.src_peerlist[i]] from My.$\mathcal{A}$;
20: Find a sensor u from My.$\mathcal{N}$, which is not in current My.$\mathcal{A}$ and its status is tentative with the highest initial priority;
21: **if** u *exists* **then** My.$\mathcal{A}$$\longleftarrow$My.$\mathcal{A} \bigcup \{u\}$;
22: **else if** *My.id* = *PKT.cluster_head* **then** Add $\mathcal{S}$ [PKT.src_id] to My.$\mathcal{C}$; Delete the PKT;
23: My.prio $\longleftarrow$ My.init_prio+$\sum_{v \in \text{My.}\mathcal{A}} v$.init_prio;

In case a cluster head is running low on battery energy, re-clustering is needed. This process can be done by sending out a re-clustering message to all the cluster members. Cluster members that receive this message switch to the initialization phase to perform a new round of clustering.

4.4 Synchronization among Cluster Heads

To perform data collection by TDMA techniques, intra-cluster time synchronization among established cluster heads should be considered. The fourth phase is to synchronize local clocks among cluster heads in a CHG by beacon messages. First, each cluster head will send out a beacon message with its initial priority and local clock information to other nodes in the CHG. Then it examines the received beacon messages to see if the priority of a beacon message is higher. If yes, it adjusts its local clock according to the timestamp of the beacon message. In our framework, such synchronization among cluster heads is only performed while SenCar is collecting data. Because data collection is not very frequent in most mobile data gathering applications, message overhead is certainly manageable within a cluster. The details of the procedure are explained in Algorithm 5.

Algorithm 5. Phase IV: Synchronization between Cluster Heads

1: **if** *My.status* $=$ *cluster_head*; **then**
2: send beacon msg with My.init_prio, My.clock, etc;
3: receive beacon msg b from other nodes in CHG;
4: **if** *b.init_prio* $>$ *My.init_prio*; **then**
5: My.clock $\longleftarrow$ b.clock;

4.5 Analysis of Load Balanced Clustering Algorithm

We have following properties about LBC algorithm.

Property 1. *Among all the cluster heads in a CHG, there is only one self-driven cluster head, and all others are peer-driven cluster heads.*

Proof. In LBC, a cluster head and its up-to-date peers form the CHG of a cluster. Clearly, it is the self-driven cluster head, and all its peers are the peer-driven cluster heads. □

Property 2. *Some clusters may have fewer than M cluster heads.*

Proof. Based on the clustering method, it is apparent that each cluster in LBC typically has a total of M cluster heads. However, some clusters may have fewer than M cluster heads. The reason can be explained as follows. To circumvent the situation that the CHGs of different clusters may share common cluster heads, sensors with tentative status always update their candidate peers once receiving status packets. For sensor s_i, once its neighbors reach their final status, if s_i is still tentative, it would update its candidate peers to see if they are the current peers. If yes, they will be expurgated from $s_i.\mathcal{A}$. We define a set $X = \{v | v \in s_i.\mathcal{N}, v \notin s_i.\mathcal{A}, v.status = tentative\}$, which represents the possible new candidate peers of s_i. s_i would choose the sensors in X with the highest initial priorities to fill the vacancy among its $M-1$ candidate peers. However, in the rare case that $X = \Phi$, s_i would have no replenishment for the vacancy. Therefore, the candidate peers of s_i could only become fewer and fewer as the update goes on. Later, if s_i happens to be a cluster head by the self-driven status transition, the size of the CHG, which is formed by s_i and its update-to-date candidate peers, would be no more than M. □

Property 3. *In LBC, a smaller M results in more clusters.*

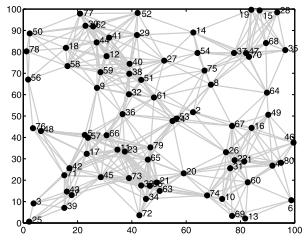
Proof. Once a sensor s_i claims to be a cluster head, its current candidate peers will also be identified to be the cluster heads immediately and their priority will be updated to the priority of s_i. s_i and all of its peers form the CHG of the corresponding cluster. Suppose s_j is one of the candidate peers of s_i (i.e., $s_j \in s_i.\mathcal{A}$). Without loss of generality, we assume that there exists another sensor in tentative status, denoted by s_k, in the neighborhood of s_j, but out of the reachable range of s_i (i.e., $s_k \in s_j.\mathcal{N}$ and $s_k \notin s_i.\mathcal{N}$). In the algorithm, if the priority of s_k is less than that of s_j, s_k will stay at tentative status through the end of the iterations in phase II. This implies that s_j essentially restrains its neighbors with lower priorities from claiming to be cluster heads. At a later time, s_k may have an opportunity to affiliate itself to s_j as a cluster member in phase III. If there are more candidate peers like s_j, more sensors in the similar situation to s_k will refrain from claiming to be cluster heads and alternatively join the current cluster as members in the periphery. Hence, the size of the cluster becomes larger with a smaller M. □

Property 4. *The total number of status packets exchanged in LBC is $O(n)$, where n is the number of sensors in the network.*

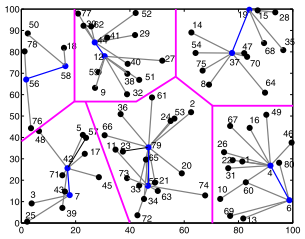
Proof. During the executions, each sensor generates at most two status packets. Specifically, on one hand, each sensor may probabilistically send a packet to indicate itself as tentative cluster head (Algorithm 2, line 4). On the other hand, each sensor will send another packet whenever it reaches its final status, either a status packet for claiming to be a cluster head or a joining message as a cluster member. Hence, when there are a total of n sensors in the network, the number of status packets sent in the network is upper-bounded by $2n$, i.e., with $O(n)$ complexity. □

Property 5. *The time complexity of LBC in the worst case is $O(n^2)$ for each sensor.*

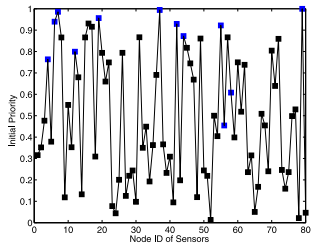
Proof. A sensor needs at most $O(n)$ time in the initialization phase to discover its neighbors and sets its priority based on the initial priorities of the neighbors. In phase II, the time cost differs from sensor to sensor. For example, a sensor, which determines to be a cluster head with the priority above τ_h, only experiences one iteration and it has no potential cluster head at all. Thus, its processing time for phase II is a constant, i.e., $O(1)$. However, in the worst case, a sensor needs to complete its maximum iterations which is determined by its node degree. And during each iteration, it is required to examine all the potential cluster heads. In such a case, the time complexity is $O(n^2)$. In phase III, a sensor takes $O(n)$ time to arbitrarily affiliate itself to a cluster head nearby. Thus, the total time complexity for a sensor in the clustering process is $O(n^2)$ in the worst case. □



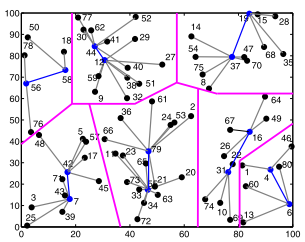
(a) Network Configuration with 80 sensors distributed in a 100×100 area.



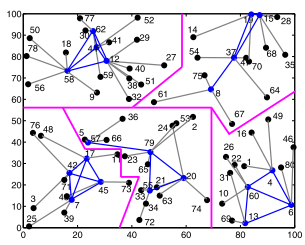
(b) LBC Clustering with $M = 2$, $\tau_h = M \times 0.9$, and $\tau_m = M \times 0.3$.



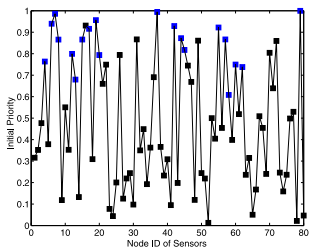
(c) Initial priority of sensors, with blue squares representing the cluster heads ($M = 2$).



(d) LBC Clustering with $M = 2$, $\tau_h = M \times 0.75$, and $\tau_m = M \times 0.4$.



(e) LBC Clustering with $M = 4$, $\tau_h = M \times 0.9$, and $\tau_m = M \times 0.3$.



(f) Initial priority of sensors, with blue squares representing the cluster heads ($M = 4$).

Fig. 4. An example of LBC with different parameter settings.

4.6 Examples of LBC Algorithm

Next, we provide an example to show the impact of some parameters on the clustering result in Fig. 4. Fig. 4a shows a random arrangement of 80 sensors on a 100×100 area. The connectivity among sensors is identified by the links between any two neighboring nodes. Fig. 4b displays the result of LBC with M set to 2. Since the priority of a sensor is the sum of its initial priority and those of its $M - 1$ candidate peers, the two thresholds, τ_h and τ_m, are proportionally set to $M \times 0.9$ and $M \times 0.3$, respectively. In Fig. 4b, sensors are self-organized into 6 clusters, each having two cluster heads shown in blue. The links between each cluster head and its members, which are shown in grey, indicate the final association pattern of the sensors. And the links shown in blue, represent the connectivity between the two cluster heads in a CHG. Fig. 4c shows the initial priority of each sensor and those of the selected cluster heads are highlighted in blue. It is observed that the final cluster heads are the ones with higher initial priorities in its proximity, which validates the requirement that the sensors in possession of higher residual energy are preferentially chosen to be cluster heads.

In Fig. 4d, we keep M unchanged and vary the settings of τ_h and τ_m to be $M \times 0.75$ and $M \times 0.4$, respectively. These relaxed thresholds imply that there is more opportunity for a sensor to immediately claim itself and its candidate peers to be cluster heads, and there is also more possibility for a sensor to quickly "retire" to be a cluster member. This helps a sensor effectively reduce the computational iterations, however, it leads to more clusters in the network. For instance, there are seven clusters formed in Fig. 4d, one more compared to Fig. 4b with more tightened thresholds. The reason for the newly-emerged cluster is because that s_{16}, which considers s_{31} as its candidate peer, promptly claims itself to be cluster head at the very beginning of its iterations for status claim. On one hand, s_{16} opportunistically excludes itself from joining the set of possible cluster heads in its proximity (i.e., $s_{16}.\mathcal{B}$) and there is also no other potential cluster head available at the starting moment. On the other hand, the priority of s_{16} is 1.8, which is the sum of the initial priorities of s_{16} and s_{31}, surpasses τ_h, which is set to 1.5. Hence, s_{16} and s_{31} successfully become cluster heads and quit the iterations beforehand. Clearly, for a given number of sensors, the relaxed thresholds lead to more cluster heads and smaller average size of clusters such that local aggregation can be beneficially shared and balanced among more nodes. However, since more clusters make themselves to get closer to each other, it necessitates more management on suppression of the inter-cluster collisions.

Finally, we vary M to investigate its impact on clustering. The results shown in Fig. 4e further validate Property 3. In the example, M is reset to 4, and τ_h and τ_m are still proportionally set to $M \times 0.9$ and $M \times 0.3$, respectively. A total of five clusters is observed when $M = 4$, less than the result in Fig. 4b with $M = 2$. Particularly, in Fig. 4b, s_{56} which is far away from cluster heads s_4 and s_{12} of the cluster nearby, claims itself and its candidate peer s_{58} to be cluster heads. In contrast, s_{58}, turning to be the peer of s_4 in Fig. 4e, successfully prevents s_{56} from attempting to be a cluster head. Thus, the two neighboring clusters in the left topmost corner in Fig. 4b merge into a larger cluster in Fig. 4e. Fig. 4f demonstrates the initial priorities of all sensors when M is set to 4 and those of selected final cluster heads are highlighted in blue. It reveals that the variation of M has no impact on the feature that LBC algorithm always chooses the sensors with higher residual energy to be the cluster heads.

5 Cluster Head Layer: Connectivity among CHGs

We now consider the cluster head layer. As aforementioned, the multiple cluster heads in a CHG coordinate among cluster members and collaborate to communicate with other CHGs. Hence, the inter-cluster communication in LBC-DDU is essentially the communication among CHGs. By employing the mobile collector, cluster heads in a CHG need not to forward data packets from other clusters. Instead, the inter-cluster transmissions are only used to forward the information of each CHG to SenCar. The CHG information will be used to optimize the moving trajectory of SenCar, which will be discussed in the next section. For CHG information forwarding, the main issue at the cluster head layer is the inter-cluster organization to ensure the connectivity among CHGs.

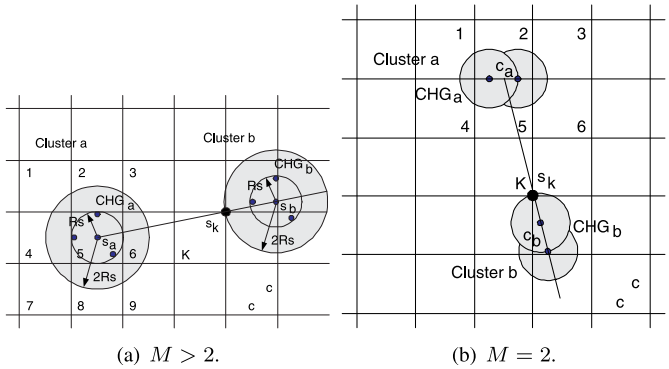
5.1 Connectivity among CHGs

The inter-cluster organization is determined by the relationship between the inter-cluster transmission range R_t and the sensor transmission range R_s. Clearly, R_t is much larger than R_s. It implies that in a traditional single-head cluster, each cluster head must greatly enhance its output power to reach other cluster heads. However, in LBC-DDU the multiple cluster heads of a CHG can mitigate this rigid demand since they can cooperate for inter-cluster transmission and relax the requirement on the individual output power. In the following, we first find the condition on R_t that ensures inter-cluster connectivity, and then discuss how the cooperation in a CHG achieves energy saving in output power.

We assume that an $l \times l$ sensor field is divided into square cells, each of which is of size $c \times c$ and $c = 2R_s$. Based on the result in [33], Ye et al. [34] showed that when n sensors are uniformly distributed and $c^2 n = kl^2 \ln l$ for some $k > 0$, each cell contains at least one sensor. When $R_t > 2(\sqrt{5}+1)R_s$, the inter-cluster connectivity can be guaranteed with single-head clustering. In a similar way, the following property gives the condition to guarantee the inter-cluster connection in LBC-DDU.

Property 6. *Under the assumption that each cell contains at least one sensor, for any cluster a and its neighboring cluster b, $\lim_{l\to\infty} P_r(\min(D(a,b)) < (\sqrt{26}+2)R_s) = 1$ when $M > 2$ and $\lim_{l\to\infty} P_r(\min(D(a,b)) < (\sqrt{17}+\frac{3}{2})R_s) = 1$ when $M = 2$, where $D(a,b)$ is the distance between clusters a and b, and $P_r(\cdot)$ represents the corresponding probability.*

Proof. Consider $M > 2$ first. In the worst case, all other clusters are far away from cluster a. Cluster a and its neighboring cluster b both have M cluster heads since if any CHG has fewer cluster heads than M, the distance between the two clusters would be shorter, which can be easily deducted from Property 3. Based on the principle of LBC, the self-driven cluster head s_a, and all its up-to-date candidate peers form the CHG of cluster a. Since these candidate peers are all in the neighborhood of s_a, all the cluster heads in the CHG of cluster a are within an area with radius R_s and centered at s_a. Since each cluster member should be covered by at least one cluster head in a CHG, the maximum coverage of cluster a is a circle area with radius $2R_s$ regardless of the value of M. The same is applicable to cluster b. Without loss of generality, we assume that cluster a is centered at a cell as shown in Fig. 5a. Given that a sensor can be located anywhere in a cell, in the worst case, sensors in cells $1 \sim 9$, the area completely or partially covered by cluster a, are all located within the range of cluster a. Consider the sensor at the closest cell outside of cluster a, say, s_k, which is located at cell k to the right of cell 6. The farthest position s_k can be from cluster a is the position of the right topmost corner of cell k. Then s_k should be within the range of cluster b. In the worst case, it is located at the periphery of the coverage area of cluster b. Hence, the maximum possible distance between the two clusters is the length of the line segment between s_a and s_b with s_a, s_k and s_b in an alignment. This distance is equal to $(\sqrt{26}+2)R_s$, which implies that within this distance there must exist at least one another cluster.



(a) $M > 2$. (b) $M = 2$.

Fig. 5. Maximum distance between two neighboring clusters when both of them have M cluster heads.

Similarly, when $M = 2$, the distance between two cluster heads in a CHG is no more than R_s and the maximum coverage area of a cluster is achieved when two cluster heads are R_s apart from each other. Fig. 5b depicts such two clusters a and b, where the shadowed area is the coverage area of each cluster. Consider cluster a. No matter where it is located and how it is oriented, it can completely or partially cover at most six cells. The worst case is that all the sensors in these six cells are in the range of cluster a. Thus, the closest sensor s_k outside of cluster a should be at the right bottommost corner of cell k, which is under cell 5. Similar to the case of $M > 2$, we can derive that the maximum possible distance between two neighboring clusters is $(\sqrt{17}+\frac{3}{2})R_s$. □

Property 7. *If inter-cluster transmission range $R_t \geq (\sqrt{26}+2)R_s$ when $M > 2$ or $R_t \geq (\sqrt{17}+\frac{3}{2})R_s$ when $M = 2$, LBC-DDU generates a connected graph among CHGs.*

This can be easily proved by contradiction.

5.2 Inter-Cluster Communications

Next, we discuss how cluster heads in a CHG collaborate for energy-efficient inter-cluster communication. We treat cluster heads in a CHG as multiple antennas both in the transmitting and receiving sides such that an equivalent MIMO system can be constructed [27]. The self-driven cluster head in a CHG can either coordinate the local information sharing at the transmitting side or act as the destination for the cooperative reception at the receiving side. Each collaborative cluster head as the transmitter encodes the transmission sequence according to a specified space-time block code (STBC) [36] to achieve spatial diversity. Compared to the single-input single-output system, it has been shown in [37] that a MIMO system with spatial diversity leads to higher reliability given the same power budget. An alternative view is that for the same receive sensitivity, MIMO systems require less transmission energy than SISO systems for the same transmission distance. Therefore, given two connected clusters, compared with the single-head structure, in which the inter-cluster transmission is equivalent to a SISO system, the multi-head structure in LBC-DDU can save energy for inter-cluster communication. In particular, the

maximum distance of two neighboring clusters is $2(\sqrt{5}+1)R_s$ [34] in single-head clustering. Thus, the required transmission power of a cluster head for such inter-cluster transmission can be expressed as follows when the free space propagation model is employed

$$P_{SHC} = \mu \cdot \frac{(4\pi)^2 L}{G_t G_r \lambda^2} \cdot \frac{\left[2(\sqrt{5}+1)R_s\right]^2}{\alpha^2}, \tag{1}$$

where μ is the given receive sensitivity, α represents the small-scale fading parameter between two cluster heads, G_t and G_r are the transmitting and receiving antenna gains, λ is the transmission wavelength and L is a system loss factor not related to propagation. In contrast, in LBC-DDU, the inter-cluster communication between two CHGs is equivalent to a MIMO transmission. Each transmitted data symbol would enjoy a diversity gain of $a_t \times a_r$, where a_t and a_r are the numbers of transmitting and receiving antennas, respectively. In the worst case, two neighboring clusters are apart as far as possible, i.e., the size of CHGs on both sides is M, and the maximum distance between two neighboring clusters is equal to the lower bound of R_t given in Property 7. Hence, the output power of each cluster head in the transmitting CHG is given by

$$P_{LBC} = \begin{cases} \mu \cdot \frac{(4\pi)^2 L}{G_t G_r \lambda^2} \cdot \frac{\left[(\sqrt{26}+2)R_s\right]^2}{\sum_{i=1}^{a_t}\sum_{j=1}^{a_r}\alpha_{ij}^2}, & a_t = a_r = M > 2 \\ \mu \cdot \frac{(4\pi)^2 L}{G_t G_r \lambda^2} \cdot \frac{\left[(\sqrt{17}+\frac{3}{2})R_s\right]^2}{\sum_{i=1}^{a_t}\sum_{j=1}^{a_r}\alpha_{ij}^2}, & a_t = a_r = M = 2. \end{cases} \tag{2}$$

where α_{ij} represents the small-scale fading parameter of the channel between the ith antenna in the transmitting CHG and the jth antenna of the receiving CHG. We assume these channels are independent and identically distributed (i.i.d).

We further define the saving ratio ρ^M as follows to evaluate the difference in the output power of a cluster head between single-head clustering and LBC

$$\rho^M = \frac{E(P_{SHC})}{E(P_{LBC})} = \begin{cases} \frac{4\left(\sqrt{5}+1\right)^2}{\left(\sqrt{26}+2\right)^2/M^2} \approx 0.83M^2, & M > 2 \\ \frac{4\left(\sqrt{5}+1\right)^2}{\left(\sqrt{17}+\frac{3}{2}\right)^2/4} \approx 5.3, & M = 2. \end{cases} \tag{3}$$

From the above discussion, we can see that the saving ratio of each cluster head is 5.3 when $M = 2$ in LBC-DDU. This ratio becomes higher with the increase of M. Thus, for long-haul inter-cluster transmissions in LBC-DDU, more cluster heads in a CHG can balance the load and save energy at each sensor.

6 SenCar Layer: Trajectory Planning

In this section, we focus on how to optimize the trajectory of SenCar for the data collection tour with the CHG information, which is referred to as the mobility control at the SenCar layer. As mentioned in Section 3, SenCar would stop at some selected polling points within each cluster to collect data from multiple cluster heads via single-hop transmissions. Thus, finding the optimal trajectory for SenCar can be reduced to finding selected polling points for each cluster and determining the sequence to visit them.

6.1 Properties of Polling Points

We consider the case that SenCar is equipped with two antennas, as it is not difficult to mount two antennas on SenCar, while it likely becomes difficult and even infeasible to mount more antennas due to the constraint on the distances between antennas to ensure independent fading. Note that each cluster head has only one antenna. The multiple antennas of SenCar, which act as the receiving antennas in data uploading, make it possible for multiple cluster heads in a CHG to transmit distinct data simultaneously. To guarantee successful decoding when SenCar receives the mixed streams, we need to limit the number of simultaneous data streams to no more than the number of receiving antennas. In other words, since SenCar is equipped with two receiving antennas, at most two cluster heads in a CHG can simultaneously send data to SenCar in a time slot. Hence, an equivalent 2×2 MIMO system for an uplink transmission is formed, which achieves spatial multiplexing gain for higher data rate. With such concurrent transmissions, data uploading time can be greatly reduced. If there are always two cluster heads that simultaneously upload their data to SenCar in each time slot, data uploading time can be cut into half in the ideal case.

In fact, when the size of a CHG is larger than 2, we have multiple choices to schedule cluster head pairs to communicate with SenCar. Each such a pair is called a *scheduling pair*. We use Π to denote all the possible scheduling options in a CHG. Without loss of generality, we assume that M is an even number. For a given schedule $\pi \in \Pi$, there are $\frac{M}{2}$ scheduling pairs. The SenCar will choose a selected polling point for each of them. When SenCar arrives at a cluster, it will visit each selected polling point, where it stops to simultaneously collect data from the two cluster heads in a scheduling pair. To collect data as fast as possible in a cluster, the following two requirements should be satisfied.

- The two cluster heads in a scheduling pair both should be covered by SenCar with the same transmission range as a sensor, i.e., R_s, when SenCar is at the selected polling point specific for this scheduling pair.
- By visiting the selected polling points in a cluster, SenCar should achieve maximum sum of the uplink MIMO capacities in the cluster.

For each polling point, we assume that SenCar has the knowledge of the IDs of sensors in the proximity within range R_s and the channel vectors between the sensors and SenCar located at the polling point. The information can be collected prior to each data collection tour. Upon receiving the CHG information which contains the IDs of the cluster heads in a CHG, for each possible schedule π, SenCar can choose a set of candidate polling points for each scheduling pair in π, each of which can cover the two cluster heads in a scheduling pair at the same time. Specifically, let $\mathcal{P}'_i$ denote such a set of candidate polling points for scheduling pair i in a cluster, where $i = 1, 2, \ldots, M/2$. Clearly, each $\mathcal{P}'_i$ is a subset of set $\mathcal{P}$ that contains all the polling points. Based on the first requirement, the selected polling point for scheduling pair i in a cluster should be chosen from $\mathcal{P}'_i$. Next, we will first find the distribution condition of the polling points to guarantee that there are always candidate polling points for choosing, and then present the criteria for selecting the

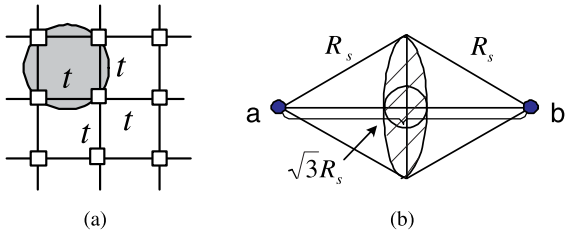


Fig. 6. (a) Distribution of polling points. (b) Candidate polling points for the scheduling pair (a,b) in a CHG should be within the shadowed area.

schedule and selected polling points, which are addressed by the second requirement.

First, we have the following property regarding the distance between two cluster heads in a scheduling pair.

Property 8. *In a CHG with M cluster heads, for any even number M, $\exists \pi \in \Pi$, for each scheduling pair (a,b) in π, such that $d_{a,b} \leq \sqrt{3}R_s$, where $d_{a,b}$ represents the distance between the two cluster heads a and b in the same scheduling pair.*

Proof. We prove the property by induction on M. As mentioned above, in a CHG, there is only one self-driven cluster head and a total of $M-1$ peer-driven cluster heads, which are denoted as CH_0, $CH_1, \ldots, CH_{M-1}$ for clarity. All of them are within a circle area with radius R_s and centered at the self-driven cluster head (i.e., CH_0). First consider the case of $M=2$. Apparently, the property holds since the distance between CH_0 and CH_1 is clearly no more than R_s. We can pair them directly. Assume that there is a feasible schedule π for some M with pair $(CH_0, CH_x) \in \pi$ where $x \in [1, M-1]$. Now we prove that schedule π for M implies a valid schedule π' for $M+2$ cluster heads. Suppose we add two new cluster heads CH_a and CH_b. If the distance between the two new heads $d_{a,b}$ is no longer than $\sqrt{3}R_s$, then there exists a valid schedule $\pi' = \pi \bigcup (a,b)$. Otherwise, we can pair (CH_x, CH_y) where $\{d_{x,y} \leq \sqrt{3}R_s \,|\, CH_y \in \{CH_a, CH_b\}\}$, and pair CH_0 with the remaining new cluster head. Based on the above discussion, we can conclude that for any even number M there exists a valid schedule satisfying $d_{a,b} \leq \sqrt{3}R_s$ for any pair (a,b). □

To successfully choose the selected polling points in a cluster, there should exist at least one possible schedule in which the sets of candidate polling points for all scheduling pairs are non-empty at the same time, i.e., $\exists \pi \in \Pi$ such that $\mathcal{P}'_i \neq \Phi$, for all the scheduling pair $i \in \pi$. This requirement imposes challenges on the distribution of polling points. We study the case that polling points are located at the intersections of grids with each polling point apart from its adjacent neighbors in horizontal and vertical directions in the same distance t, as plotted in Fig. 6a. We have the following property on t to satisfy the above requirement.

Property 9. *If polling points are uniformly distributed as depicted in Fig. 6a, for a CHG with M cluster heads, when $t \leq \sqrt{2}(1-\frac{\sqrt{3}}{2})R_s$, regardless of the value of M, $\exists \pi \in \Pi$, for each scheduling pair $(a,b) \in \pi$, $\sum_{p \in \mathcal{P}} P_r\big(d_{a,p} \leq R_s, d_{b,p} \leq R_s\big) \neq 0$, where $\mathcal{P}$ denotes the set of all polling points, $d_{a,p}$ and $d_{b,p}$ are the distances between cluster heads a and b in a scheduling pair and the polling point p, respectively.*

Proof. Under the above specified distribution of polling points, regardless of the orientation of grids, there is at least one polling point located in a circle area with radius $\frac{\sqrt{2}}{2}t$. For example, in Fig. 6a, there are 4 polling points distributed in such an area, which is highlighted in shadow. Moreover, it is known from Property 8 that there exist some schedules, in which the distance between two cluster heads in any scheduling pair is upper bounded by $\sqrt{3}R_s$. Consider the worst case that there is a scheduling pair (a,b) with $d_{a,b} = \sqrt{3}R_s$ (see Fig. 6b). A candidate polling point p for (a,b) should be within the transmission range of cluster heads a and b simultaneously. The line shadowed area in Fig. 6b, which is the intersection of transmission areas of a and b, indicates the possible distribution region of candidate polling points for (a,b). It is clear that when $d_{a,b}$ is shorter, the area of the region would be larger, which corresponds to the lower distribution density requirement on polling points. Hence, considering the case with $d_{a,b} = \sqrt{3}R_s$ is sufficient for the proof of the property. It is shown in Fig. 6b that there exists an inscribed circle with radius $(1-\frac{\sqrt{3}}{2})R_s$ inside the line shadowed area. If $t = \sqrt{2}(1-\frac{\sqrt{3}}{2})R_s$, substituting R_s with the expression of t, the radius of the inscribed circle in the line shadowed area is equal to $\frac{\sqrt{2}}{2}t$. As mentioned earlier, there is at least one polling point located in such an area. Thus there always exist some candidate polling points for scheduling pair (a,b), i.e., $\sum_{p \in \mathcal{P}} P_r(d_{a,p} \leq R_s, d_{b,p} \leq R_s) \neq 0$. In other words, when $t \leq \sqrt{2}(1-\frac{\sqrt{3}}{2})R_s$, there exist some schedules in which the set of candidate polling points for each scheduling pair is always non-empty. □

6.2 MU-MIMO Uploading

We jointly consider the selections of the schedule pattern and selected polling points for the corresponding scheduling pairs, aiming at achieving the maximum sum of MIMO uplink capacity in a cluster. We assume that SenCar utilizes the minimum mean square error receiver with successive interference cancellation (MMSE-SIC) as the receiving structure for each MIMO data uploading. Based on this receiver, the capacity of a 2×2 MIMO uplink between a scheduling pair (a,b) and SenCar located at a selected polling point $\triangle$ can be expressed as follows.

$$C^{\triangle}_{(a,b)} = \log\left(1 + \frac{P_t \|\mathbf{h}_a\|^2}{N_0 \mathbf{I}_2 + P_t \|\mathbf{h}_b\|^2}\right) + \log\left(1 + \frac{P_t \|\mathbf{h}_b\|^2}{N_0}\right), \quad (4)$$

where $\mathbf{h}_a$ and $\mathbf{h}_b$ are two 2×1 channel vectors between cluster heads a and b and SenCar at $\triangle$, respectively, P_t is the output power of a sensor for transmission range R_s, and N_0 is the variance of the back-ground Gaussian noise. The MMSE-SIC receiver first decodes the information from a, treating the signals of b as the interference. Then, it cancels the signal part of a from the received signals. The remaining signal part of b only has to contend with the background Gaussian noise.

Accordingly, the criteria for the schedule and the selected polling point for each corresponding scheduling pair in a

cluster is given by

$$[\pi, \triangle_1, \triangle_2, \ldots, \triangle_{\frac{M}{2}}] = \underset{\pi \in \Pi, \triangle_i \in \mathcal{P}'_i}{\arg\max} \left(\sum_{(a,b)\in\pi} C^{\triangle_i}_{(a,b)} \right), \quad (5)$$

where π is a specified schedule, scheduling pair $i \in \pi$ consists of cluster heads a and b, $\triangle_i$ and $\mathcal{P}'_i$ are the selected polling point and the set of candidate polling points for scheduling pair i, respectively, and $C^{\triangle_i}_{(a,b)}$ is the achieved 2×2 MIMO uplink capacity for scheduling pair i when SenCar is positioned at $\triangle_i$.

Once the selected polling points for each cluster are chosen, SenCar can finally determine its trajectory. The moving time on the trajectory can be reduced by a proper visiting sequence of selected polling points. Since SenCar departs from the data sink and also needs to return the collected data to it, the trajectory of SenCar is a route that visits each selected polling point once. This is the well-known traveling salesman problem (TSP). Since SenCar has the knowledge about the locations of polling points, it can utilize an approximate or heuristic algorithm for the TSP problem to find the shortest moving trajectory among selected polling points, e.g., the nearest neighbor algorithm [35].

6.3 Data Collection with Time Constraints

In this section, we further consider the case when there are time constraints on data messages. In practice, it is common for some emergent data messages to be delivered within a specified deadline. If the deadline has expired and the message is yet to arrive at the destination, it would carry less value and cause performance degradation. In [25], mobile data collection with dynamic deadline was considered and an earliest deadline first algorithm was proposed. In their solution, the mobile collector would visit the nodes with messages of the earliest deadline. Here, we extend and adapt their solutions to the clustered network. Our method is described in the following.

First, the cluster heads collect data messages and calculate a deadline by averaging all the deadlines from messages in the cluster. All the clusters then forward their deadline information to SenCar. The SenCar selects the cluster with the earliest average deadline and moves to the polling point to collect data via MU-MIMO transmissions. After SenCar finishes data gathering, it checks to see whether collecting data from the next polling point would cause any violations of deadline in its buffer. If yes, it immediately moves back to the data sink to upload buffered data and resumes data collection in the same way. By prioritizing messages with earlier deadlines, SenCar would do its best to avoid missing deadlines.

7 Performance Evaluations

In this section, we evaluate the performance of our framework and compare it with other schemes. Since the main focus of this paper is to explore different choices of data collection schemes, for fair comparison, we assume all the schemes are implemented under the same duty-cycling MAC strategy. The first scheme for comparison is to relay messages to a static data sink in multi-hops and we call it *Relay Routing*. Since nodes with higher battery energy provide more robustness and error immunity, sensors select the next hop neighbor with the highest residual energy while forwarding messages to the sink. Once some nodes on a routing path consume too much energy, an alternative route will be chosen to circumvent these nodes. In this way, the relay routing method can provide load balance among nodes along the routing path. The second scheme to compare is based on *Collection Tree Protocol*, [6]. In CTP, the expected number of transmission (ETX) is used as a routing metric and the route with a lower ETX takes precedence over routes with higher ETX. For simplicity, we assume ETX is proportional to transmission distances between nodes. This assumption is reasonable since using fixed power for longer transmission distance would cause attenuated receiving power and potentially increase error probability and expected number of transmissions. Based on this metric, we establish a collection tree rooted at the static data sink at the origin (0, 0). Each node forwards messages along the path with the lowest ETX towards the sink. Any broken links caused by nodes depleted battery energy would lead to large ETX and are avoided in routings.

Furthermore, we introduce two schemes that organize nodes into clusters and relay messages to the static data sink by multi-hop transmissions. The third scheme to compare is *clustered SISO* in which sensors form into clusters with a single cluster head. Data messages are converged at the cluster heads and relayed to the static data sink in multi-hops by SISO communications. By the same token, we can extend the third scheme to *clustered MIMO* which allows MIMO communications between the cluster heads according to [27]. On the other hand, we also include the fifth scheme called *mobile SISO* which is the traditional way for mobile data gathering, where SenCar stops in each cluster for data collection and uploads data to the sink after all the clusters have been visited. The last scheme is our proposed LBC-DDU framework (denoted by *mobile MIMO* for clarity) that elects multiple cluster heads to enable MU-MIMO uploading to SenCar in each cluster.

We develop a simulator in MATLAB and discuss the parameter settings in the following. A total of n sensors are randomly scattered in an $l \times l$ field. The static data sink is located at $(0, 0)$. There are a total of n_p polling points uniformly distributed in the field. The sensor transmission range R_s is $40\ m$. Each sensor has installed an AA battery of 1,500 mAh. To estimate energy consumptions for SISO, we use the model presented in [10], i.e., $e_t = (e_1 d_r^{\alpha} + e_0) l_p$, where e_t is the energy consumption while transmitting a message of l_p bits, d_r is the transmission range, e_1 is the loss coefficient per bit, α is the path loss exponent and e_0 is the excessive energy consumed on sensing, coding, modulations, etc.[1] For MIMO uploading between cluster heads and SenCar, we utilize the results for 2×2 MIMO in [27]. When $d_r = 40$ m, the energy consumption per bit is 0.6×10^{-5} J/bit. Each sensor holds $5{,}120$ bytes sensing data. Each data message is 10 bytes and each control message is 1 byte (overhead). Therefore, energy consumptions per data message for SISO and MIMO are 1.28 and $0.48\,mJ$,

1. $d_r = 40$m, $e_0 = 45 \times 10^{-9}$ J/bit, $e_1 = 10 \times 10^{-9}$ J/bit, $\alpha = 2$.

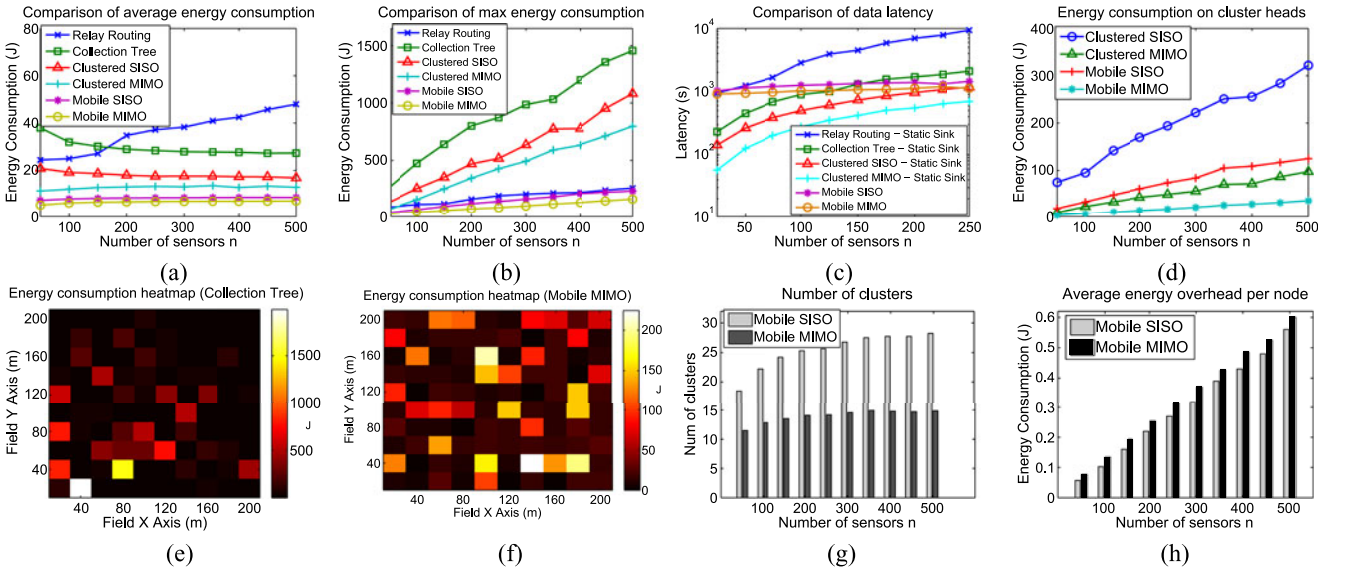


Fig. 7. Performance comparisons of different data gathering schemes when $M = 2$. (a) Average energy consumptions per node. (b) Maximum energy consumption. (c) Data latency. (d) Average energy consumption for each cluster head. (e) Energy consumption heatmap of Collection Tree scheme. (f) Energy consumption heatmap of Mobile MIMO scheme. (g) Number of clusters. (h) Average energy overhead per node.

respectively. The circuit energy consumption of MIMO uploading itself is modeled in [27]. Since the receiver is SenCar, we only consider energy overhead on the transmitters' side, which are the cluster head groups. From [27], we obtain that extra energy consumption incurred on cluster heads using MIMO is, $P_e = P_{DAC} + P_{mix} + P_{filt}$, in which $P_{DAC} = 1.95\,mW$ according to Texas Instrument's DAC128S085 datasheet [38], $P_{mix} = 30.3\,mW$ and $P_{filt} = 2.5\,mW$ according to [29]. We obtained that for transmitting a data message, $0.03\,mJ$ extra power is consumed in MIMO circuitry (5.8 percent energy overhead from the circuitry). The transmission bandwidth is $100\,Kbps$ and the speed of SenCar is $1\,m/s$. Each performance point is the average of the results in 200 simulation experiments.

7.1 Comparison of Energy Consumptions and Latency

First, we compare the average energy consumption for each sensor and the maximum energy consumption in the network. We set $l = 250\,m$, $n_p = 400$, and $M = 2$ (at most two cluster heads for each cluster) and vary n from 50 to 500. Note that when $n = 50$, network connectivity cannot be guaranteed all the time for multi-hop transmission with a static sink. The results here are only the average of the connected networks in the experiments. However, the mobile schemes can work well not only in connected networks but also in disconnected networks, since the mobile collector acts as virtual links to connect the separated subnetworks.

Fig. 7 a compares the average energy consumption per node. We can see that our mobile MIMO scheme results in the least energy consumption on sensor nodes, whereas the methods that transmit messages through multi-hop relay to the static data sink result in at least twice more energy on each node. Fig. 7b further presents the maximum energy consumption in the network. The network lifetime usually lasts until the first node depletes its energy. It is intuitive that schemes with lower maximum energy consumption would have longer network lifetime. Fig. 7b shows that mobile MIMO has the least maximum energy consumptions and curves for the schemes that utilize multi-hop relaying to the static data sink (Relay Routing, Collection Tree and Clustered SISO/MIMO) climb much faster and higher than mobile MIMO scheme. When we increase the size of the network, network lifetime of these schemes would deteriorate because more relayed traffic needs to traverse the congested region near the data sink.

Second, we illustrate data latency by Fig. 7c. Here, we define data latency as the time duration for the data sink to gather all the sensing data in the field. For mobile schemes, data latency is equivalent to the time duration of a data collection tour which comprises of the moving time and data transmission time. In Fig. 7c, lower latency is achieved with clustered SISO/MIMO compared to Relay Routing or Collection Tree methods. This is because that nodes are organized into clusters and routing burden is decomposed into smaller tasks by different clusters. Relay routing has the highest latency because choosing node with the highest energy as the next hop may not lead to the shortest path. On the other hand, mobile SISO/MIMO has a little higher latency than clustered SISO/MIMO and Collection Tree methods due to SenCar's moving time. However, note that when $n > 300$, the latency of the Collection Tree method exceeds mobile SISO/MIMO and the slopes for multi-hop relaying to the static sink are much steeper than mobile SISO/MIMO. This is because that though mobility incurs extra moving time, single-hop transmissions for both data aggregation and uploading save time in routing significantly, whereas multi-hop traffic relay to the static sink may not scale well when the number of nodes increases. Finally, the advantage of mobile MIMO comparing to mobile SISO is observed by saving 20 percent time in total. This is due to the fact of simultaneous data uploading to SenCar at the polling points in the mobile MIMO approach.

7.2 Load Balance and Energy Distributions

We also evaluate load balance on cluster heads in Fig. 7d. Cluster heads consume more energy than other nodes due

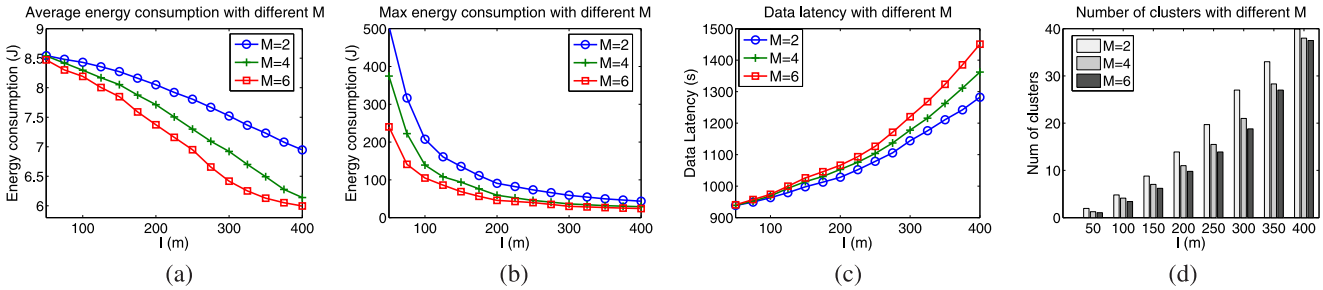


Fig. 8. Performance comparison of proposed framework with different M. (a) Average energy consumption per node versus l. (b) Maximum energy consumption versus l. (c) Data latency versus l. (d) Number of clusters versus l.

to data aggregations and forwarding. Reducing energy consumptions on cluster heads improves network longevity and robustness. In Fig. 7d, we can see that first both MIMO approaches outperform SISO, and mobile MIMO achieves good load balance among cluster heads with the lowest average energy consumption by saving over 60 percent energy compared to the mobile SISO. In sum, the mobile scheme outperforms multi-hop relay to the static sink because no inter-cluster data forwarding is needed and the spatial diversity achieved by adopting MIMO further reduces energy consumption of cluster heads.

To justify our choice of mobile data collections, we also compare the geographical energy distribution between Collection Tree and mobile MIMO. For demonstration purposes, we set $n = 200$ and draw the heat map of energy consumption in Figs. 7e and 7f. We observe that more energy is consumed with the Collection Tree method especially on nodes near the data sink represented by the bright spots in Fig. 7e. These nodes may become congested bottlenecks and jeopardize the operation of the network. Although mechanisms in Collection Tree can find a better route by adapting to ETX metrics, congestion is inevitable due to the physical locations of these nodes. In contrast, the mobile MIMO method results in much less energy consumption and even distribution across the sensing field in Fig. 7f.

7.3 Number of Clusters and Energy Overhead

Fig. 7g shows the number of clusters generated by mobile SISO and MIMO. We can see that the mobile MIMO typically yields fewer clusters than mobile SISO. Note that the average energy consumptions of mobile SISO and MIMO become indistinguishable as n increases. This is because that although fewer clusters are observed with mobile MIMO, more cluster heads are generated for each cluster. Thus, the total number of cluster heads in the two schemes turns to be comparable, which is actually a dominant factor determining energy consumptions.

Finally, we compare the energy overhead with mobile SISO and MIMO, and illustrate the energy consumption by MIMO uploading itself in Fig. 7h. For the mobile approaches, overhead is mainly comprised of status and synchronization messages in clustering, messages notifying SenCar of cluster locations and circuit energy consumption if MIMO uploading is adopted. First, the number of status messages exchanged is proved to be upper bounded by $2n$ in Property 4 in Section 5, where n is the number of nodes, and synchronization messages between cluster heads are only propagated within each cluster. Second, after clusters are formed, each cluster needs to forward the locations of cluster heads to SenCar resided at the origin for finding the polling points. Third, if MIMO uploading is adopted, there is excessive energy consumption of $0.03\,mJ$ per data message.

By considering these aspects, we plot the energy overhead in Fig. 7h. First of all, it is observed that energy overhead results in less than 6 percent over the total energy consumptions. Then, the gap between mobile SISO and MIMO represents extra energy consumed in the MIMO circuitry is small. It indicates that compared to the energy overhead on clustering and notifying SenCar of cluster information, the energy consumed by MIMO circuity itself is negligible.

7.4 Impact of Maximum Number of Cluster Heads in Each Cluster

In this section, we evaluate the impact of the maximum number of cluster heads in each cluster, M, on energy consumptions, latency and number of clusters in Fig. 8. We plot the performance of mobile MIMO with different M when l varies from 50 to $400\,m$ and $n = 200$. We fix the interval distance t between a polling point and its adjacent neighbors in horizontal and vertical directions at about $20\,m$, which means that n_p varies from 16 to 441 with different settings of l.

Fig. 8a shows that the average energy consumption declines as l increases in all cases. This is because that more clusters would be formed when sensors become sparsely distributed as indicated in Fig. 8d. It is also noticed that a larger M leads to less energy consumptions. For example, when $l = 200\,m$, energy consumption with $M = 4$ is 35 percent less than the case of $M = 2$. This result is intuitive since cluster heads perform more transmissions than other nodes. When M increases, there are more cluster heads in a cluster to share the workload. Fig. 8b shows the maximum energy consumption in the network. Since more cluster heads can directly upload their data to SenCar without any relay, the case with a larger M results in a slightly less energy consumptions. For example, when $l = 300\,m$, maximum energy consumption with $M = 6$ is 15 percent less than the case with $M = 2$.

In Fig. 8c, it is demonstrated that a larger M also leads to longer data latency. The reason is that more selected polling points need to be visited, which leads to a longer moving trajectory. For instance, when $l = 400\,m$, the data latency in the case of $M = 6$ and $M = 4$ is 14 and 7 percent longer than the case of $M = 2$, respectively. Fig. 8d shows the number of clusters formed with different M. It further

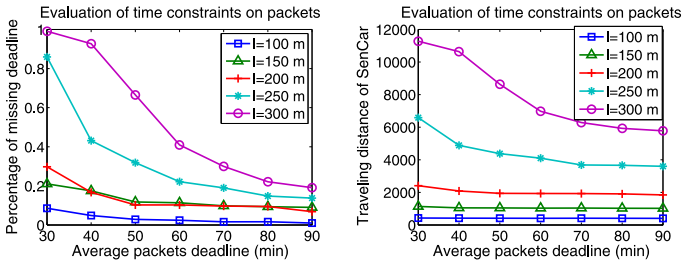


Fig. 9. Evaluation of data collection with time constraints. (a) Percentage of data messages that miss the deadline. (b) Impact of time constraints on traveling cost of SenCar.

validates that a larger M typically leads to fewer clusters. However, it is also noticed that the difference becomes less evident when l becomes larger. This is because that many clusters formed under this condition actually have fewer cluster heads than M since sensors become sparsely distributed such that the controlling impact of M on the cluster size is not fully extracted.

7.5 Data Collection with Time Constraints

In this section, we demonstrate our proposed framework when data messages have time constraints to be delivered. The percentage of data messages that miss their deadlines and the impact of time constraints on traveling cost of SenCar are shown in Fig. 9. To examine the effectiveness of the proposed algorithm in Section 6.3, we set the message deadline to be uniformly randomly distributed over $[0, X]$ and change X from 60 to 180 mins. Therefore, the mean of deadline is from 30 to 90 mins. The number of nodes n is set to 200 and the side length of sensing field l varies from 100 to 300 with an increment of 50. In Fig. 9a we can see that given a short average deadline and a large field ($l = 300\,m$, mean deadline equals 30 mins), and almost all the messages would miss their deadlines. This is because that the moving time of SenCar to traverse all the polling points exceeds most of the deadlines. Once we relax the deadline constraints (mean deadline equals 40-90 mins), the percentage of missing deadlines drops fast. We also observe that when l is between 100 to 200 m, the algorithm is able to maintain the percentage of missing deadlines within 20 percent for most cases.

To meet dynamic message deadlines, extra moving cost on SenCar is observed. This is because that the cluster with an earlier message deadline may be far from SenCar's location. We present the traveling cost of SenCar for different cases in Fig. 9b and compare it with the traveling cost computed by the nearest neighbor heuristic algorithm [35] in Table 2 (no time constraint on messages). When $l = 300\,m$ and mean deadline equals 90 mins, the moving cost on SenCar is over $5{,}000$ m compared to $1{,}846$ m without time constraints. When $l = 100\,m$ and mean deadline equals 90 mins, the moving cost on SenCar is 404 m compared to 347 m without time constraints. The results indicate that higher cost on SenCar is needed with a larger field size. To further reduce traveling cost on SenCar while guaranteeing to meet all the time constraints, multiple SenCars are needed to partition the data collection tour and many existing solutions for the Vehicle Routing Problems with Time Windows [39] can be adapted to optimize the data collection routes.

TABLE 2
Traveling Cost without Time Constraints

Side length l (m)	100	150	200	250	300
Moving cost (m)	347	557	974	1,511	1,846

8 Conclusions and Future Works

In this paper, we have proposed the LBC-DDU framework for mobile data collection in a WSN. It consists of sensor layer, cluster head layer and SenCar layer. It employs distributed load balanced clustering for sensor self-organization, adopts collaborative inter-cluster communication for energy-efficient transmissions among CHGs, uses dual data uploading for fast data collection, and optimizes SenCar's mobility to fully enjoy the benefits of MU-MIMO. Our performance study demonstrates the effectiveness of the proposed framework. The results show that LBC-DDU can greatly reduce energy consumptions by alleviating routing burdens on nodes and balancing workload among cluster heads, which achieves 20 percent less data collection time compared to SISO mobile data gathering and over 60 percent energy saving on cluster heads. We have also justified the energy overhead and explored the results with different numbers of cluster heads in the framework.

Finally, we would like to point out that there are some interesting problems that may be studied in our future work. The first problem is how to find polling points and compatible pairs for each cluster. A discretization scheme should be developed to partition the continuous space to locate the optimal polling point for each cluster. Then finding the compatible pairs becomes a matching problem to achieve optimal overall spatial diversity. The second problem is how to schedule MIMO uploading from multiple clusters. An algorithm that adapts to the current MIMO-based transmission scheduling algorithms should be studied in future.

Acknowledgments

This work was supported in part by the US NSF grant numbers ECCS-0801438 and ECCS-1307576 and US ARO grant number W911NF-09-1-0154.

References

[1] B. Krishnamachari, *Networking Wireless Sensors*. Cambridge, U.K.: Cambridge Univ. Press, Dec. 2005.
[2] R. Shorey, A. Ananda, M. C. Chan, and W. T. Ooi, *Mobile, Wireless, Sensor Networks*. Piscataway, NJ, USA: IEEE Press, Mar. 2006.
[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
[4] W. C. Cheng, C. Chou, L. Golubchik, S. Khuller, and Y. C. Wan, "A coordinated data collection approach: Design, evaluation, and comparison," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 10, pp. 2004–2018, Dec. 2004.
[5] K. Xu, H. Hassanein, G. Takahara, and Q. Wang, "Relay node deployment strategies in heterogeneous wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 2, pp. 145–159, Feb. 2010.
[6] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. 7th ACM Conf. Embedded Netw. Sensor Syst.*, 2009, pp. 1–14.
[7] E. Lee, S. Park, F. Yu, and S.-H. Kim, "Data gathering mechanism with local sink in geographic routing for wireless sensor networks," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1433–1441, Aug. 2010.

[8] Y. Wu, Z. Mao, S. Fahmy, and N. Shroff, "Constructing maximum-lifetime data-gathering forests in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1571–1584, Oct. 2010.
[9] X. Tang and J. Xu, "Adaptive data collection strategies for lifetime-constrained wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 6, pp. 721–7314, Jun. 2008.
[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–660, Oct. 2002.
[11] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," in *IEEE Conf. Comput. Commun.*, pp. 366–379, 2004.
[12] D. Gong, Y. Yang, and Z. Pan, "Energy-efficient clustering in lossy wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 73, no. 9, pp. 1323–1336, Sep. 2013.
[13] A. Amis, R. Prakash, D. Huynh, and T. Vuong, "Max-min d-cluster formation in wireless ad hoc networks," in *Proc. IEEE Conf. Comput. Commun.*, Mar. 2000, pp. 32–41.
[14] A. Manjeshwar and D. P. Agrawal, "Teen: A routing protocol for enhanced efficiency in wireless sensor networks," in *Proc. 15th Int. IEEE Parallel Distrib. Process. Symp.*, Apr. 2001, pp. 2009–2015.
[15] Z. Zhang, M. Ma, and Y. Yang, "Energy efficient multi-hop polling in clusters of two-layered heterogeneous sensor networks," *IEEE Trans. Comput.*, vol. 57. no. 2, pp. 231–245, Feb. 2008.
[16] M. Ma and Y. Yang, "SenCar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 10, pp. 1476–1488, Oct. 2007.
[17] B. Gedik, L. Liu, and P. S. Yu, "ASAP: An adaptive sampling approach to data collection in sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 12, pp. 1766–1783, Dec. 2007.
[18] C. Liu, K. Wu, and J. Pei, "An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 7, pp. 1010–1023, Jul. 2007.
[19] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a three-tier architecture for sparse sensor networks," *Elsevier Ad Hoc Netw. J.*, vol. 1, pp. 215–233, Sep. 2003.
[20] D. Jea, A. A. Somasundara, and M. B. Srivastava, "Multiple controlled mobile elements (data mules) for data collection in sensor networks," in *Proc. IEEE/ACM Int. Conf. Distrib. Comput. Sensor Syst.*, Jun. 2005, pp. 244–257.
[21] M. Ma, Y. Yang, and M. Zhao, "Tour planning for mobile data gathering mechanisms in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1472–1483, May 2013.
[22] M. Zhao and Y. Yang, "Bounded relay hop mobile data gathering in wireless sensor networks," *IEEE Trans. Comput.*, vol. 61, no. 2, pp. 265–271, Feb. 2012.
[23] M. Zhao, M. Ma, and Y. Yang, "Mobile data gathering with space-division multiple access in wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2008, pp. 1283–1291.
[24] M. Zhao, M. Ma, and Y. Yang, "Efficient data gathering with mobile collectors and space-division multiple access technique in wireless sensor networks," *IEEE Trans. Comput.*, vol. 60, no. 3, pp. 400–417, Mar. 2011.
[25] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava,, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Proc. 25th IEEE Int. Real-Time Syst. Symp.*, Dec. 2004, pp. 296–305.
[26] W. Ajib and D. Haccoun, "An overview of scheduling algorithms in MIMO-based fourth-generation wireless systems," *IEEE Netw.*, vol. 19, no. 5, Sep./Oct. 2005, pp. 43–48.
[27] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-efficiency of MIMO and cooperative MIMO techniques in sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1089–1098, Aug. 2004.
[28] S. Jayaweera, "Virtual MIMO-based cooperative communication for energy-constrained wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 5, no. 5, pp. 984–989, May 2006.
[29] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-constrained modulation optimization," *IEEE Trans. Wireless Commun.*, vol. 4, no. 5, pp. 2349–2360, Sep. 2005.
[30] I. Rhee, A. Warrier, J. Min, and X. Song, "DRAND: Distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Proc. 7th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2006, pp. 190–201.
[31] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Netw.*, vol. 16, no. 4, pp. 985–997, May 2010.
[32] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: A hybrid MAC for wireless sensor networks," in *Proc. 3rd ACM Int. Conf. Embedded Netw. Sensor Syst.*, 2005, pp. 90–101.
[33] D. M. Blough and P. Santi, "Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks," in *Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw.*, 2002, pp. 183–192.
[34] F. Ye, G. Zhong, S. Lu, and L. Zhang, "PEAS: A robust energy conserving protocol for long-lived sensor networks," in *Proc. 23rd IEEE Int. Conf. Distrib. Comput. Syst.*, 2003, pp. 28–37.
[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.
[36] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block codes for orthogonal designs," *IEEE Trans. Info. Theory*, vol. 45, no. 5, pp. 1456–1467, Jul. 1999.
[37] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, May 2005.
[38] [Online]. (2013). Available: http://www.ti.com/product/DAC128S085/technicaldocuments
[39] M. M. Solomon, "Algorithm for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.

Miao Zhao received the BEng and MEng degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, and the PhD degree in electrical engineering, Stony Brook University, New York, in 2010. She is currently a member of technical staff with Multimedia Networking Lab of Huawei Technologies, Bridgewater, New Jersey. Her research interests include cross-layer design and optimization in wireless networks, and performance evaluation of protocols and algorithms. She is a member of the IEEE.

Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a professor of computer engineering and computer science at Stony Brook University, New York, and the director of Communications and Devices Division at New York State Center of Excellence in Wireless and Information Technology (CEWIT). Her research interests include wireless networks, data center networks, optical networks, and parallel and distributed computing systems. She has published more than 300 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She is currently the associate editor-in-chief for the *IEEE Transactions on Computers* and an associate editor for the *Journal of Parallel and Distributed Computing*. She has served as an associate editor for the *IEEE Transactions on Computers* and the *IEEE Transactions on Parallel and Distributed Systems*. She has served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is a fellow of the IEEE.

Cong Wang received the BEng degree of information engineering from the Chinese University of Hong Kong in 2008, and the MS degree in electrical engineering from Columbia University, New York, in 2009. He is currently working toward the PhD degree at the Department of Electrical and Computer Engineering, Stony Brook University, New York. His research interests include wireless sensor networks, performance evaluation of network protocols and algorithms.