

Mobile Data Collector Assignment and Scheduling for Minimizing Data Delay in Partitioned Wireless Sensor Networks

Izzet F. Senturk^(✉) and Kemal Akkaya

Southern Illinois University, Carbondale, IL 62901, USA
{isenturk,kemal}@cs.siu.edu

Abstract. Mobile Data Collectors (MDCs) can be employed to provide intermittent connectivity in partitioned Wireless Sensor Networks (WSNs). Typically, an MDC serves multiple partitions to collect their sensed data and relay it to the sink node if the sink is in one of the served partitions. If not, then the data needs to be passed to another MDC serving the sink partition at a certain rendezvous location. This causes an extra waiting time for the MDCs that reach the meeting points before others. In this paper, we propose a scheduling algorithm to reduce the rendezvous waiting delay based on the tour lengths. To completely eliminate this waiting delay, we also present a solution with multiple sinks. In this approach, the partitions of the WSN are clustered using p -center optimization and Voronoi cells. An MDC is then assigned for touring each cluster's partitions. In this way, the dependency of MDCs to relay their data to other MDCs is eliminated and the tour lengths of MDCs are balanced which minimizes the end-to-end delay significantly.

Keywords: WSNs · Mobile data collector · Delay · Scheduling · Clustering · Energy consumption · Minimum tour length

1 Introduction

The exposure of sensor nodes to challenging environmental conditions leaves WSNs susceptible to network partitioning. Such partitioning impairs the network operations and data collection and thus recovery schemes need to be pursued. In order to restore data communication with the sink, one solution is to deploy relay nodes (RNs) in the damaged regions so that the RNs can fill the gaps between the partitions and re-connect the WSN [1,2]. However, the number of available RNs may not always be sufficient to connect all the partitions. In such a case, if RNs have the ability to move, they can be employed as mobile data collectors (MDCs) to provide intermittent connectivity between the partitions and the sink(s) [3,4].

Recently, several research challenges regarding the exploitation of MDCs have been tackled. The main focus of the studies was to reduce the tour lengths of

MDCs when they are visiting each partition to collect the sensor data. Since, typically multiple MDCs are available, multiple tours exist and the partitions involved in these tours are pre-arranged based on a clustering algorithm [5,6]. To have load-balanced tour lengths (i.e., to reduce the maximum tour length and data latency), the existing clustering algorithms strived to provide balanced clusters [5,7]. In such clusters, MDCs collect data from all of the partitions and then relay it to the sink node, if in that partition, or to another MDC so that it will relay data to the sink node. This introduces the problem of rendezvous waiting which has not been considered in the previous studies. The implicit assumption in such studies is that the sensors within the partition can hold the data collected from an MDC indefinitely and then forward it to another MDC once it is nearby. Since sensors have limited buffers, this may not be feasible when multiple MDCs are dumping their data to the same sensors. Therefore, clustering should be done by also considering the impact of possible rendezvous waiting so as to minimize the overall data latency.

In this paper, we first demonstrate the impact of clustering on data delay with a single sink. We introduce rendezvous waiting delay as one of the major factors when determining the total data delay when several partitions exist. We then propose a scheduling algorithm to alleviate the waiting time by checking the possible locations which have potential to introduce rendezvous delay. Specifically, the waiting is eliminated if another tour is possible within the rendezvous waiting time. In this way, some of the partitions' data will be transferred without waiting for the data coming from other partitions which will minimize the average delay in the network.

Since this scheduling does not eliminate the rendezvous waiting completely, we later introduce a clustering approach which eventually eliminates the waiting time completely. The idea is to designate multiple sinks to collect data to eliminate the dependency of an MDC on other MDCs to relay packets. This raises the problem of load-balanced clustering of partitions based on a given number of sinks. As part of this problem, we tackle two sub-problems: First, determining the right locations for the sinks among the partitions; second, creating the clusters of partitions around these sinks.

For determining the sink locations, we follow the idea that the locations should be apart from each other as much as possible as used in p -center optimization problem [8]. Note that during this process each partition is assumed to be represented as a single entity whose location is the centroid of all the sensors within that partition. Once the sink locations are determined, we assign the sensor nodes that are closest to these locations as the sinks to collect data from its cluster.

For clustering, we propose an approach based on Voronoi cells. We define an area of proximity (Voronoi cell) for the sinks where the partitions within the Voronoi cell will have the minimum distance to the corresponding sink compared to other sinks. Once the clusters are formed, each cluster is served with an MDC to collect the data from the partitions in the cluster and deliver to the sink affiliated with that cluster. We use Traveling Salesman Problem (TSP)-based modeling while planning the tours [5] for the MDCs.

The proposed approaches are evaluated under a variety of conditions via extensive simulations. The simulation results indicate that end-to-end delay decreases with our proposed scheduling approach. The delay is further reduced upon applying the proposed clustering schemes.

The rest of the paper is organized as follows. Related work is summarized in Sect. 2. The assumptions and the problem definition are provided in Sect. 3. The proposed scheduling approach for single sink is explained in Sect. 4. Clustering for multiple sinks is explained in Sect. 5. The performance of the proposed approaches are evaluated in Sect. 6. The paper is concluded in Sect. 7.

2 Related Work

The problem of relay placement is well-studied in the context of WSNs. A majority of these works considered stationary relays and focused on the placement of these relays to guarantee connectivity [2, 9, 10]. Later, a number of works among these also handled the cases where there is not enough number of relays and thus some of the relays should acts as mobile relays [5, 7]. In those works, the main goal for the mobile relays was to reduce the tour length along with some constraints. A final category of works among these focused on the sole use of mobile relays, referred to as MDCs, assuming that none of the relays can stay stationary. This is the category we focus in this paper.

This third category is similar to k -TSP problem where k travelers travel n cities [11]. In the context of WSNs, this has been studied as data collection from n sensors for a single mobile robot with different constraints [12]. In addition, this problem has been applied to connectivity restoration in disjoint WSNs. For instance, in a recent study [6] investigated k -TSP solution assuming that k MDCs will be connecting several partitions. The authors propose a polynomial time heuristic for interconnecting Disjoint Segments with k MDCs with the single goal of minimizing the maximum tour length of all the MDCs. A similar work is done in [13]. In their solution, the employed mobile RNs keep moving in the network based on a certain policy to provide the intermittent connectivity over a time period instead of continuous connectivity. The main contribution of the scheme is to mathematically model the movement of the mobile RNs bridging fragments as a closed queuing network and achieve steady state results for distributing the mobile RNs in the network.

The above solutions focus on minimization of tour length which is different than our work in this paper. Our work can be considered as a previous phase before the tours are computed. Specifically, we focus on an assignment problem where the partitions are clustered in such a way that for each cluster an MDC is assigned. Our goal is to do this assignment so that the delay for data collection is minimized. To the best of our knowledge, this is a novel problem which has not been studied in the context of connectivity restoration in WSNs.

Our problem also includes scheduling of the tours so that there will be minimal rendezvous waiting when MDCs are relaying data from other MDCs. Rendezvous planning and mobile scheduling have been studied in different

problems and contexts in the past. For instance, [14,15] strive to meet delay requirements for sensor data when a mobile is collecting data from them. They propose to place rendezvous nodes to different locations of the network so that every sensor can relay their data to these nodes and the mobile just tours these rendezvous nodes to reduce the overall delay. This type of rendezvous planning is very different than ours since in fact there is no meeting of two MDCs. This is similar to clustering the network and assigning a sink to each of them which is different than our problem.

Per mobile scheduling, there are several related works that perform scheduling for other purposes. For instance, in [16], the goal is to schedule the visits of a mobile to collect data with a given deadline so that the sensors relaying data to the mobile will not buffer overflow. In this case, there may be different sampling rates and thus the mobile may have to visit a sensor more than the others which makes the problem different than TSP. The authors present both an ILP formulation and heuristics to solve this problem. Another partitioning based heuristic for the same problem was presented in [17]. In our work, there is no deadline but we do scheduling to minimize the waiting times of MDCs.

3 Preliminaries

3.1 Assumptions

We assume a WSN with disjoint partitions detached from the rest of the network due to initial deployment or node failures. Initially, we assume a single sink which collects data from all the sensors within this WSN. The nodes in the WSN are assumed to be stationary and therefore the network topology cannot be restructured without any external intervention. We assume the availability of mobile RN(s) referred to as MDCs equipped with a memory large enough to store the data of several sensor nodes. We also assume the application area is obstacle-free.

Connectivity of a partitioned network can be restored by deploying RNs to locations identified by an RN placement algorithm. However, excessive number of RNs may be required to guarantee connectivity and we assume the lack of sufficient RNs available. Thus, we employ RNs as MDCs to collect data from partitions and deliver to sink so that an intermittent connection is provided to ensure the sustainability of data communication. We use an RN placement algorithm, CIST [18], to determine the minimum RN count to guarantee connectivity and the interface sensors for the partitions (i.e., the sensors which will communicate with MDCs). Based on the minimum number, we assume MDC count (i.e., it should be less than this number). A sample demonstration of the CIST showing the interface sensors is illustrated in Fig. 1.

3.2 Problem Definition

Our problem can be more formally defined as follows: “Given a set of n nodes $\in N$ which form $k \geq 2$ disjoint partitions, our goal is to restore network connectivity

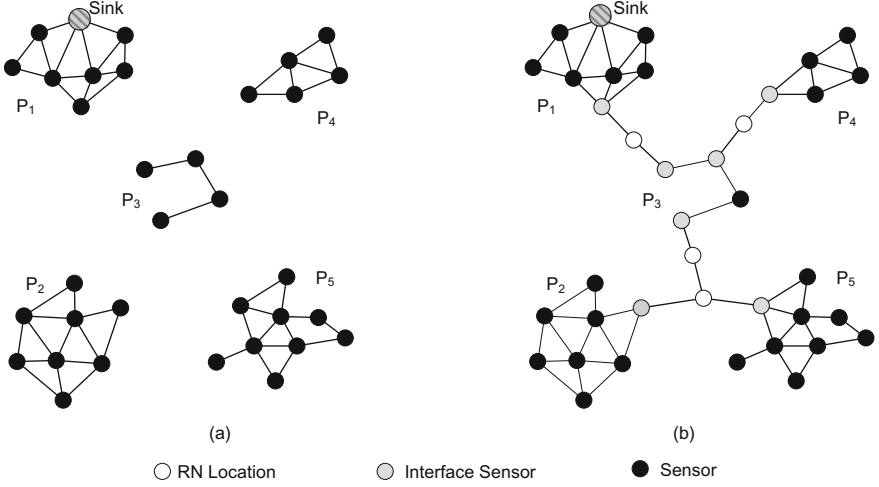


Fig. 1. A sample illustration for a partitioned WSN (a) and the planned 4 RN locations that will reconnect the WSN as a result of CIST [18] algorithm (b). Gray nodes are interface sensors obtained by CIST.

by employing $m > 0$ (mobile) RNs in the area while minimizing the average data delay from sensors to the sink.”

The data delay is defined as follows: Once the connectivity is restored, each node, n_i , will transmit its packets to the destination (sink) through the upstream path P_i . P_i comprises a set of node IDs representing hops to reach the sink. Thus, for a delay, d_j , incurred at each hop, total delay of a packet sent from n_i to sink can be stated as $\sum_{j=1}^{|P_i|} d_j$ where $|P_i|$ denotes the number of hops in P_i .

As part of d_j , we consider three different types of delay where a packet may be subject to. If the hop j is not served with an MDC, delay can be simply defined as: $d_j = R_j$ where R_j denotes the transmission delay. On the other hand, if the hop j is served with an MDC RN_j , then delay on hop j can be stated as $d_j = T_j + W_j + R_j$ where T_j denotes the touring delay of RN_j and W_j denotes the waiting delay of the immediate MDCs exchanging data with RN_j .

As part of the solution, we investigate two approaches: First, we will assume a single sink node and propose a scheduling algorithm to minimize the waiting delay. Second, we will assume multiple sinks and propose a clustering algorithm to further reduce the data latency.

4 Rendezvous Scheduling of MDCs for Reduced Delay

4.1 Background on MDC Tours

In this section, we briefly explain how the partitions are clustered and the tours are computed when the network has a single sink. Depending on the number of available MDCs, say m , we create m clusters of partitions and assign each cluster

an MDC. The clustering starts with pairing of partitions and then merging them until the number of clusters matches with m [7]. The MDC is expected to tour the partitions within a cluster and collect data from interface sensors. The tour of the MDC is computed using Traveling Salesman Problem (TSP) [19] as done in [5]. An example of clustering and tours is provided in Fig. 2.

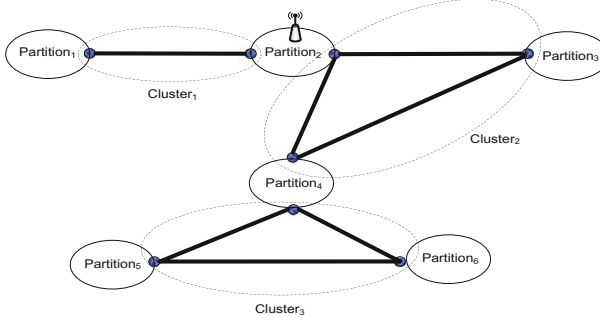


Fig. 2. The creation of clusters based on the number of MDCs. Note that a partition may have multiple interface sensors to communicate with an MDC.

Since each MDC collects data from all the partitions within a cluster, it needs to forward them to the sink node. If the sink is not in one of the partitions within the cluster, then the MDC should forward the collected data to another MDC (e.g., MDC of *cluster3* should forward data to MDC of *cluster2* that has the sink). Since the MDC may not dump all its collected data to a sensor in the partition due to buffer constraints of the sensors, this may introduce some rendezvous waiting time which has not been considered in the previous work. Especially in WSNs with several clusters, waiting time for MDCs can be a dominating factor in the total delay.

Next, we analyze this delay as part of the whole data delay and then propose a scheduling algorithm to reduce it.

4.2 End-to-End Delay with a Single Sink

Given that rendezvous waiting can be a big issue impacting the delay, in this subsection we will describe a computation mechanism for determining the waiting time of MDCs and assess its impact on the overall delay.

Each MDC will wait at certain meeting points for the MDCs touring in the neighboring clusters. This is referred to as “wait-for-all children” approach to synchronize all the data going to the sink node. A meeting point resides at the intersection of two or more clusters. Since the whole set of partitions will be connected, clusters will intersect at certain partitions. For instance, if two clusters contain sensor nodes from the same partition, then the tours will intersect at the meeting points which represent the locations to contact the interface sensors of

the common partition. Thus, meeting point does not represent a point where two MDCs reside at the same time, but rather two different positions possibly apart from each other with a distance larger than their transmission range (see Fig. 3). The waiting time of an MDC at a meeting point depends on the immediate predecessor MDCs (i.e., children) and the immediate successor MDC (i.e., the MDC that is closer to sink). Multiple MDCs can constitute the set of children but only one MDC will be the successor.

A sample scenario depicting meeting points and children is represented in Fig. 3. In this example, partition P_4 is the common partition of clusters $C_a \cap C_b \cap C_c \cap C_d$. Based on the direction towards the sink, it can be observed that MDC_a , MDC_b and MDC_c are the predecessor (children) MDCs of MDC_d . On the other hand, partition P_5 is the common partition of clusters $C_d \cap C_e$ and MDC_e is the successor MDC of MDC_d . Thus, we can identify two meeting points for MDC_d , one for P_4 and one for P_5 . The interface nodes are already obtained during clustering and the locations to contact P_4 and P_5 are determined while forming the tour of MDC_d . Now we detail how to calculate the rendezvous waiting time of MDC_d at meeting points m_2 and m_3 .

In order to initiate the data exchange at m_2 , MDC_a , MDC_b and MDC_c should be present on m_1 while MDC_d is present on m_2 . They can then forward their data via the sensors in the partition to MDC_d . For the waiting time of MDC_d at m_2 , in the worst case, we can assume that MDC_d will have to wait the children with the longest tour length (e.g., MDC_a) to finish a complete tour. Similarly, in order to initiate the data exchange at m_3 , MDC_e should be present at m_4 while MDC_d is at m_3 . For the waiting time of MDC_d at m_3 , in the worst case, we can assume that MDC_d will have to wait for MDC_e to finish a complete tour. Thus, the total waiting time of MDC_d at m_2 and m_3 can be expressed as in Eq. 1 where t is the number of predecessor MDCs.

$$W^{MDC_d} = W_{m_2}^{MDC_d} + W_{m_3}^{MDC_d} = \max_{i=1}^t T^{MDC_i} + T^{MDC_e} \quad (1)$$

Since none of the predecessor MDCs of MDC_d has a child, no waiting delay occurs for MDC_a , MDC_b and MDC_c . Otherwise, waiting time of these MDCs should also have been considered in a recursive way. Similarly, since MDC_e is connected to sink, no further waiting delay affects MDC_d at m_3 . Again, otherwise, MDC_d would be subject to waiting delays of MDC_e in a recursive way.

4.3 Rendezvous Scheduling for Reduced Delay

In order to reduce the rendezvous waiting times, we strive to exploit the time period between these waiting times. Specifically, if the MDCs start moving at certain positions, waiting delay can be decreased.

We will explain the idea on an example. Let's revisit the sample topology in Fig. 3. Let the tour lengths of MDC_a , MDC_b , MDC_c , and MDC_d be $l + x$, $l + y$, $l - z$ and l respectively where $x, y, z < l$. Assume that the first data

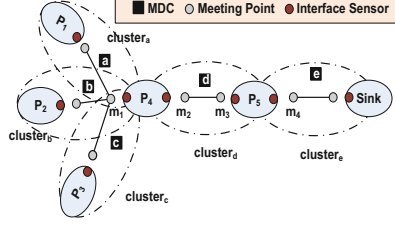


Fig. 3. A sample illustration of the occurrence of the waiting time in the collaborating MDCs

exchange is completed among MDCs at meeting points m_1 and m_2 . The next round for MDC_d will be after finishing a complete tour and returning back to m_2 . Due to its shorter tour length, MDC_c will be present at m_1 but it will take another $\max(x, y)$ time for both MDC_a and MDC_b to also complete their tours. Therefore, MDC_c can check the feasibility of another tour rather than waiting there indefinitely for MDC_a and MDC_b . If $l - z$ length can be toured within the waiting time, MDC_c will prefer this tour to increase the data sampling rate from the sensors in P_3 .

Therefore, we can express the waiting delay for MDC_d as the summation of waiting delay at m_2 and m_3 as in Eq. 2:

$$W^{MDC_d} = W_{m_2}^{MDC_d} + W_{m_3}^{MDC_d} = (\max - T^{MDC_d}) + (T^{MDC_e} - T^{MDC_d}) \quad (2)$$

where $\max = \max_{i=1}^t T^{MDC_i}$ and t is the number of preceding MDCs. Note that we take 0 as the waiting time if $\max < T^{MDC_d}$ or $T^{MDC_e} < T^{MDC_d}$. This is the difference from Eq. 1 which may provide the opportunity to have 0 waiting time for some of the MDCs.

Once an MDC computes the waiting delays at each meeting point using the above equations (the algorithms for the computation are skipped due to space constraints), it can determine its schedule by comparing its touring delay with the waiting delays.

We propose a scheduling algorithm to initiate a tour if it has enough time to complete the tour before the corresponding waiting delay expires. It is important to identify the direction of the tour. If the direction is towards the sink, waiting delay of the successor MDCs should be considered. Otherwise, waiting delay of the predecessor MDCs should be considered. The pseudo-code for this proposed algorithm is given in Algorithm 1.

Specifically, in lines 1 and 2, waiting delay to be incurred due to the predecessor and the successor MDCs are computed. In line 3, the tour info of the MDC is obtained. The direction of the tour is determined in line 4. The tour may be towards the sink partition or the opposite direction. If the direction is towards the sink, in order to initiate another tour, waiting delay to be incurred due to successor MDCs should be greater than the sum of touring delay and the delay to be incurred due to predecessor MDCs. This condition is checked in lines 4–7.

Algorithm 1. `schedule(MDC m)`

```

1:  $sDelay = \text{getSuccessorWD}(m)$   $\triangleright$  get successor delay
2:  $pDelay = \text{getPredecessorWD}(m)$   $\triangleright$  get predecessor delay
3:  $t = \text{getTour}(m)$   $\triangleright$  get tour info of m
4: if  $\text{directionTowardsSink}(t) == \text{true}$  then
5:   if  $sDelay \geq TD(T) + pDelay$  then
6:      $\text{makeTour}(t)$ 
7:   end if
8: else
9:   if  $pDelay \geq TD(T) + sDelay$  then
10:     $\text{makeTour}(t)$ 
11:   end if
12: end if

```

In the opposite direction, waiting delay to be incurred due to predecessor MDCs should be greater than the sum of touring delay and the delay to be incurred due to successor MDCs to initiate another tour. This condition is checked in lines 8–12.

5 Delay-Aware Clustering with Multiple Sinks

5.1 Motivation

The phenomenon of rendezvous waiting emerges when the clusters are not formed in a star-like topology where the sink resides in the center. Such a clustering requires the deployment of the sink at the central area of the topology. However, we assume a random deployment where the sink can be located at any location. While it is possible to form all the tours to include the sink and hence avoid rendezvous waiting, this will cause to increase the tour lengths significantly which unnecessarily increases the touring delay. Moreover, as shown in the previous section, even though it is possible to perform scheduling to reduce rendezvous waiting time, it is not possible to completely eliminate it (as will be shown in the Experiments) with a single sink and thus it remains as a major factor of long end-to-end delays.

Consequently, in this section, we present an alternative solution assuming multiple sinks are available in the WSN. Considering the fact that the waiting delay emerges due to dependency of MDCs on other MDCs, we provide an algorithm to cluster the partitions in such a way that the data of different clusters will be delivered to sinks simultaneously without any interactions with MDCs of other clusters.

5.2 Approach Overview

The approach eventually clusters the partitions but it first needs to determine the sink locations. Based on the available MDC count, sink locations are identified.

For m MDCs, m sinks are selected in the network. Considering the location of the selected sinks, the network is clustered into m clusters. The data of the nodes in each cluster is collected by the sink of the cluster. An MDC is assigned for the cluster so that the partitions in the cluster are toured by this MDC. Since the MDCs are only responsible from their clusters, demand for data exchange between the MDCs of different clusters is avoided.

Determining Sink Locations. The goal in sink selection is to minimize the maximum distance from partitions to the sinks. This can be modeled as a p -center problem where one wants to build p warehouses in different cities. Given that there are n cities, the goal is to minimize the maximum distance between the cities and the warehouses. In our case, the warehouses are the sinks and the cities are the partitions.

Following this idea, the first sink is located based on the maximum distance to the rest of the partitions in the network. The distance to partitions is based on the centroid location (i.e., location of average x and y in the partition) of the partition. The first sink is added to the set of sinks. We determine the rest of the sinks based on the maximum distance to this set of sinks. Once the locations are identified, the closest sensors to these locations are designated as sinks. After identifying the sinks, we proceed with the clustering of the network.

Clustering. Our proposed clustering approach exploits Voronoi cells while determining the partitions of each cluster. A Voronoi cell keeps the points whose distance to cell center is less than or equal to its distance to any other center in the region [20]. Based on the given sink locations, the network is divided into Voronoi cells as shown in Fig. 4a. For any point in the cell, this solution ensures that the corresponding sink has the minimum distance to the point. Each cell is regarded as a cluster and the partitions residing in the cell is assigned to the corresponding cluster. If nodes of a partition resides in multiple clusters, we identify the cluster to assign based on the location of the partition's centroid location. Note that in this approach there is a possibility of having clusters

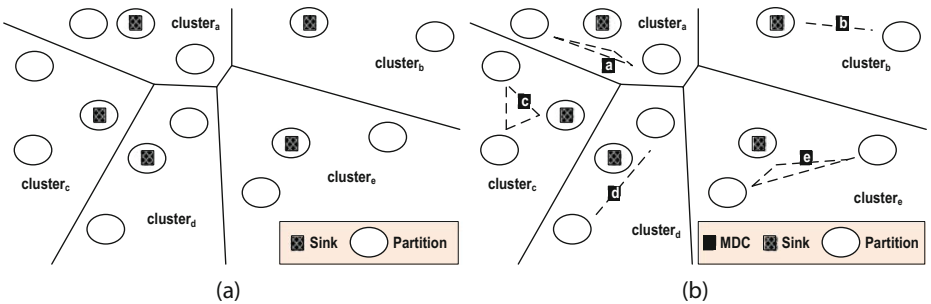


Fig. 4. Clustering of partitions using Voronoi cells (a). Forming tours in the clusters (b).

without any partitions. In that case, an MDC will not be required and it can be used by other clusters. Extra MDCs are made available to the clusters with longer tour lengths. It should be noted that multiple MDCs in the same cluster do not cause rendezvous waiting delay since they would be placed apart from each other equally and they can initiate movement at the same time.

Once the partitions are clustered, an MDC is assigned for each cluster to tour the partitions so that the dependency of MDCs as well as the waiting delay is avoided. A sample illustration of the MDC tours in the clusters is represented in Fig. 4b.

6 Experimental Evaluation

6.1 Experiment Setup

In order to evaluate the approaches, network topologies are created with disjoint partitions. A Java-based simulator is used to form the topologies and run the simulations. To deploy the nodes, an application area of 800×800 m is considered. Transmission range of the nodes is set to 50 m. The approaches are tested under a varying number of sensors nodes, number of partitions and number of MDCs. For significance, 30 different topologies are created for each test case and the average is reported.

6.2 Performance Metric and Baselines

We consider average end-to-end delay of all the nodes as the main performance metric in this paper. End-to-end delay is regarded as the time required for a packet to be transmitted across a network from source to destination. Minimizing this time is crucial for applications in general but specifically for real-time applications. This delay includes the transmission delay, touring delay and waiting delays as mentioned before. A constant value of 0.1 ms is used as the transmission delay (including the propagation delay) between two hops. For touring delay, we assume a delay of 1 s for each 10 m of movement.

For the single sink case, we considered the approach without any scheduling as one of the baselines. This one, shown as *No Scheduling (NS)* in the graphs, is used to assess the impact of scheduling. Our proposed approach is shown as *Rendezvous Aware Scheduling (RAS)* in the graphs. As the baseline approach, we compare the delay to that of [6] when there is a single sink and k MDCs. This approach exploits k -TSP problem and strives to minimize the tour lengths. We implemented this approach with and without scheduling. The version with the scheduling is shown as *IDM- k -MDC-RAS* while the one without the scheduling is *IDM- k -MDC-NS*.

For the multiple sink case, our approach is represented as *Voronoi-based Clustering-Scheduled (VBCS)*. The approach was run under different number of sinks and compared to the best case with the single sink nodes which is *RAS*.

6.3 Experiment Results and Analysis

We conducted experiments to evaluate the approaches in terms of end-to-end delay. The results are presented for solutions assuming single sink and multiple sinks separately.

Experiment Results with Single Sink. In this subsection, we provide the experiment results of the solutions assuming single sink in the network. The results are depicted in Fig. 5a, b, and Fig. 6 for varying number of partitions, nodes, and MDC count respectively.

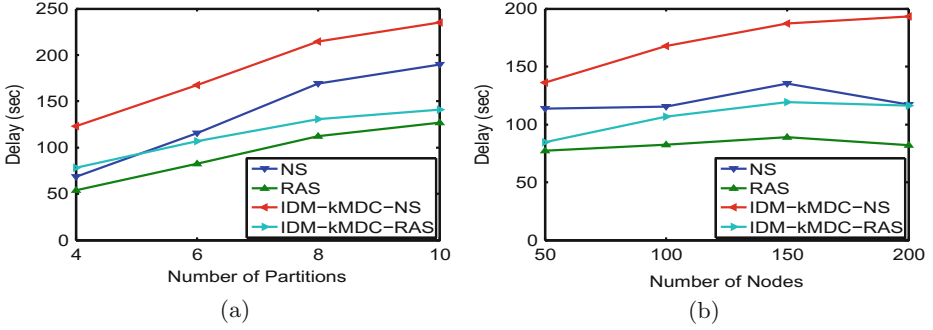


Fig. 5. End-to-end delay of sensors for varying partition count. Node count: 100, MDC count: 2 (a). End-to-end delay of sensors for varying node count. Partition count: 6, MDC count: 2 (b).

Figure 5a portrays the results for varying number of partitions when the number of nodes and available MDC count is fixed. It can be observed from Fig. 5a that delay is increasing for all of the approaches when the number of partitions is increased. This can be explained as follows: For a partitioned network with a single sink, only the sensors within the same partition with the sink can deliver their data without using an MDC. The sensors in other partitions require an MDC to collect their data. For a constant number of sensors, if the number of partitions is increased, the number of sensors in the sink partition will be reduced and thus more sensors will require an MDC to send their data. As a result, not only the tour lengths and MDC touring delays but also the rendezvous waiting delay increases.

It can also be observed from Fig. 5a that the proposed *RAS* significantly outperform *NS* and *IDM - kMDC* for both versions. This is due to the application of the scheduling algorithm in *RAS* and allowing additional tours when long waiting times is expected. Both *RAS* and *NS* outperform the respective versions of *IDM - kMDC*. This can be attributed to the underlying clustering algorithm used. Our clustering algorithm is based on [7] and provides a better load balancing than *IDM - kMDC* and thus this reduces the rendezvous waiting times.

Figure 5b presents the results for varying number of nodes when the number of partitions and the available MDC count is fixed. It can be observed from Fig. 5b that delay is increasing when scheduling is not applied. When applied, the delay increases up to 150 nodes and then starts declining. Even though we provide the average delay per node, the increase in delay when the number of nodes is increased can be attributed to longer paths to reach the sink. However, upon reaching the threshold as in the application of scheduling, the delay starts to decline. This is due to the shorter distance between the partitions with the increased node density which causes shorter tour lengths. Our approach outperforms $IDM - kMDC - RAS$ again due to same reasons explained above.

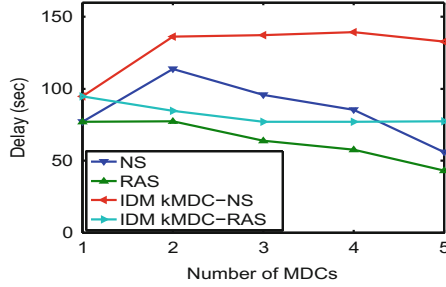


Fig. 6. End-to-end delay of sensors for varying MDC count. Node count: 50, Partition count: 6.

Figure 6 portrays the results for varying MDC count when the number of nodes and partition count is fixed. It can be observed from Fig. 6 that delay declines when the number of available MDCs is increased and scheduling is applied. On the other hand, delay increases initially and then declines when scheduling is not applied. Albeit the initial increase/decrease, the change in delay of $IDM - kMDC$ for both versions is not very significant afterwards. Additional MDCs provide shorter tour lengths and less delay for our approaches. However, due to the poor clustering and not applying the scheduling, waiting delay of more MDCs can increase in $IDM - kMDC$ which affects the overall end-to-end delay adversely.

Experiment Results with Multiple Sinks. In this subsection, we provide the experiment results of the proposed approach assuming multiple sinks in the network. The results are depicted in Fig. 7a, b, and Fig. 8 for varying number of partitions, nodes, and MDC count respectively.

Figure 7a portrays the results for varying number of partitions when the number of nodes is fixed. It can be observed from Fig. 7a that delay increases when the number of partitions is increased due to the reasons explained earlier.

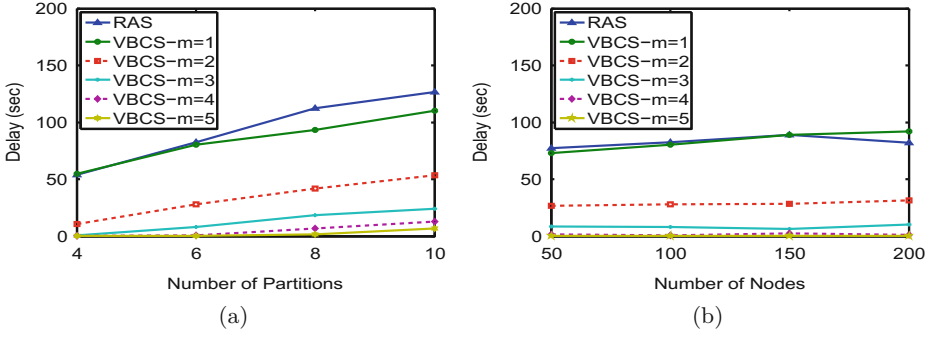


Fig. 7. End-to-end delay of sensors for varying partition count. Node count: 100, MDC count for *RAS*: 2 (a). End-to-end delay of sensors for varying node count. Partition count: 6, MDC count for *RAS*: 2 (b).

We also observe that delay reduces with the increased MDC count for *VBCS*. *VBCS* with a single MDC outperforms *RAS* due to a better clustering based on a good sink location and Voronoi cells. Recall that for *RAS*, the sink location is picked randomly.

Figure 7b presents the results for varying number of nodes when the number of partitions is fixed. It can be observed from Fig. 7b that delay is increasing slightly for the *VBCS* with MDC count 1 when the number of nodes is increased. On the other hand, delay is almost constant with multiple MDCs when the proposed approach is applied in networks with different node densities. The slight increase in delay can be attributed to the longer paths to reach the sink. Since the partition count is constant, delay is not affected by the node density significantly when multiple MDCs are available. *RAS* performs slightly better than the *VBCS* with 1 MDC. However, with 2 or more MDCs *VBCS* is still much better.

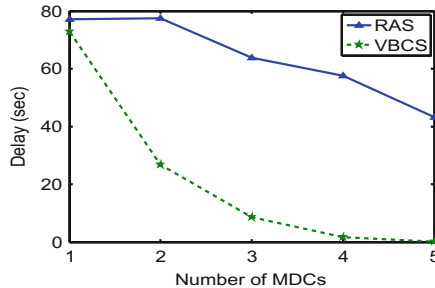


Fig. 8. End-to-end delay of sensors for varying MDC count. Node count: 50, partition count: 6.

Figure 8 portrays the results for varying MDC count when the number of nodes and partition count is fixed. It can be observed from Fig. 8 that delay declines for both approaches when the number of available MDCs is increased. The decline in delay for *RAS* can be attributed to the decrease of the touring delays by the availability of MDCs in higher count. On the other hand, for *VBCS*, the number of clusters increases with the increased MDC count. Increasing the cluster count decreases the average distance to sink which helps to minimize the touring delays. Another advantage is avoiding rendezvous waiting delay which is avoided due to delivery of data to different sinks.

7 Conclusion

In this paper, we tackled the problem of rendezvous waiting in disjoint WSNs where connectivity is provided via MDCs in an intermittent manner. We first demonstrated that with the existing approaches where multiple MDCs are used to tour the partitions in disjoint WSNs, there will be situations where MDCs need to wait for other MDCs to receive their data and forward it to the sink. We proposed a scheduling algorithm to alleviate the problem based on the tour lengths of neighboring clusters' MDCs. However, since this does not completely eliminate the waiting problem, we proposed another approach with the use of multiple sinks. This approach aimed at parallelizing the data collection within each cluster and thus waiting is not experienced by the MDCs. We propose mechanism to identify the right sink locations and cluster the partitions around these locations such that every sink serves an MDC. The determination of sink locations was based on p -center optimization and the clustering followed the idea of Voronoi cells.

The evaluation of the proposed approaches revealed that rendezvous scheduling reduces the waiting delay significantly and thus our approach outperforms the existing approaches where rendezvous waiting is not considered. The experimentation with multiple sinks has also shown the significant impact of eliminating rendezvous waiting and minimizing the total delay.

Acknowledgment. This work is supported by US National Science Foundation under the grant number CNS 1018404. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Lloyd, E.L., Xue, G.: Relay node placement in wireless sensor networks. *IEEE Trans. Comput.* **56**(1), 134–138 (2007)
2. Younis, M., Akkaya, K.: Strategies and techniques for node placement in wireless sensor networks: a survey. *Elsevier Ad-Hoc Netw. J* **6**(4), 621–655 (2008)

3. Wang, W., Srinivasan, V., Chu, K.C.: Using mobile relays to prolong the lifetime of wireless sensor networks. In: *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (Mobicom05)*, Cologne, Germany, August 2005
4. Ma, M., Yang, Y.: Data gathering in wireless sensor networks with mobile collectors. In: *IEEE International Symposium on Parallel and Distributed Processing, 2008, IPDPS 2008*, pp. 1–9 (2008)
5. Abbas, A., Younis, M.: Interconnecting disjoint network segments using a mix of stationary and mobile nodes. In: *IEEE Conference on Local Computer Networks (LCN 2012)*, Clearwater, FL (to appear)
6. Senel, F., Younis, M.: Optimized interconnection of disjoint wireless sensor network segments using k mobile data collectors. In: *IEEE International Conference on Communications (ICC'12)*, Ottawa, Canada, June 2012
7. Senturk, I., Akkaya, K., Senel, F., Younis, M.: Connectivity restoration in disjoint wireless sensor networks using limited number of mobile relays. In: *2013 IEEE International Conference on Communications (ICC)* (to appear)
8. Mladenović, N., Labbé, M., Hansen, P.: Solving the p-center problem with tabu search and variable neighborhood search. *Networks* **42**(1), 48–64 (2003)
9. Cheng, X., Du, D.-Z., Wang, L., Xu, B.: Relay sensor placement in wireless sensor networks. *Wirel. Netw.* **14**(3), 347–355 (2008)
10. Zhang, W., Xue, G., Misra, S.: Fault-tolerant relay node placement in wireless sensor networks: problems and algorithms. In: *INFOCOM 2007, 26th IEEE International Conference on Computer Communications*, pp. 1649–1657. IEEE, May 2007
11. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, pp. 216–227. IEEE Computer Society, Washington (1976)
12. Bhadauria, D., Isler, V.: Data gathering tours for mobile robots. In: *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS'09, pp. 3868–3873. IEEE Press, Piscataway (2009)
13. Almasaeid, H., Kamal, A.E.: Data delivery in fragmented wireless sensor networks using mobile agents. In: *10th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Chania, Greece, October 2007
14. Xing, G., Wang, T., Jia, W., Li, M.: Rendezvous design algorithms for wireless sensor networks with a mobile base station. In: *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 231–240. ACM (2008)
15. Xing, G., Wang, T., Xie, Z., Jia, W.: Rendezvous planning in wireless sensor networks with mobile elements. *IEEE Trans. Mob. Comput.* **7**(12), 1430–1443 (2008)
16. Somasundara, A.A., Ramamoorthy, A., Srivastava, M.B.: Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In: *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, ser. RTSS '04, pp. 296–305. IEEE Computer Society, Washington (2004). <http://dx.doi.org/10.1109/REAL.2004.31> [Online]
17. Gu, Y., Bozdag, D., Ekici, E., Ozguner, F., Lee, C.G.: Partitioning based mobile element scheduling in wireless sensor networks. In: *IEEE SECON*, pp. 386–395. Citeseer (2005)
18. Senel, F., Younis, M.: Optimized connectivity restoration in a partitioned wireless sensor network. In: *2011 IEEE Global Telecommunications Conference (GLOBE-COM 2011)*, pp. 1–5, December 2011

19. Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton University Press, Princeton (2007)
20. Bash, B.A., Desnoyers, P.: Exact distributed voronoi cell computation in sensor networks. In: Abdelzaher, T.F., Guibas, L.J., Welsh, M. (eds.) Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN 2007, Cambridge, Massachusetts, USA, 25–27 April 2007, pp. 236–243. ACM (2007)

Ad Hoc Networks

5th International ICST Conference, ADHOCNETS 2013,
Barcelona, Spain, October 2013, Revised Selected
Papers

Sherif, M.H.; Mellouk, A.; Li, J.; Bellavista, P. (Eds.)

2014, X, 229 p., Softcover

ISBN: 978-3-319-04104-9