Journal of Network and Computer Applications 66 (2016) 151–165

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Restoring connectivity in a resource constrained WSN

Yatish K. Joshi, Mohamed Younis *

Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD 21250, United States

ARTICLE INFO

Article history:
Received 5 November 2015
Accepted 11 March 2016
Available online 15 March 2016

网络恢复的重要性

Keywords:
Topology repair
Fault recovery
Connectivity restoration
Fault-tolerance
Wireless Sensor Networks
Mobile data collectors

ABSTRACT

Wireless Sensor Networks (WSNs) in applications like battlefield surveillance or environmental monitoring are usually deployed in inhospitable environments, in which their constituent nodes are susceptible to an increased risk of failure due to hazardous operating conditions or adversary attacks. In these scenarios it is possible for multiple nodes to fail at the same time and partition the WSN into disjoint segments. Such loss of connectivity may cause service disruptions and render the WSN useless. Given the critical role a WSN plays and the fact that deployment of additional nodes may be infeasible, the WSN must have the ability to self-heal and restore connectivity by utilizing surviving resources. In this paper we present a distributed *R*esource *C*onstrained *R*ecovery (RCR) approach that reconnects a network partitioned into disjoint segments by strategically repositioning nodes to act as relays. In case the number of surviving relocatable nodes are insufficient to form a stable inter-segment topology, some of them are employed as mobile data collectors with optimized tours to reduce data latency. The performance of RCR is validated through mathematical analysis and simulation.

1. Introduction

Due to their cost advantage, ease of use, rapid deployment and round the clock operation, the use of Wireless Sensor Networks (WSN) is increasingly prevalent in applications designed to operate in harsh environments like battlefields, border surveillance, rain forests etc. In these applications, the WSN is setup in an ad-hoc manner, with sensor nodes being deployed aerially in the area of interest, and once on the ground coordinating with one another to form an interconnected network to carry out application specific tasks. In such setup and operation scenarios nodes have an increased risk of failure due to the harsh environmental conditions or due to enemy action. For example, bombing or missile strikes in a battlefield or a forest fire could render a large collection of nodes inoperable and partition the network into disjoint segments. Given the unattended operation and the infeasibility of timely deployment of additional nodes, the WSN must be able to recover from collocated node failures autonomously in order to restore connectivity and resume its service.

重构方法分类

Multiple strategies have been pursued to tolerate multi node failures in WSNs (Younis et al., 2014). Published schemes can be classified broadly into proactive and reactive strategies. Proactive strategies involve careful node-placement wherein node positions are determined prior to network deployment and then leveraged to provision redundancy by forming a k-connected topology (Han et al., 2007; Li and Hou, 2004; Wang et al., 2003) or using redundant nodes as backups (Chen et al., 2001; Wang et al., 2005). Although a proactive strategy can lower the risk of network partitioning, it does not provide sufficient mitigation in ad-hoc network formation scenarios since the node positions cannot be accurately determined at time of deployment and a large scale failure may damage some nodes and all their backups.

Reactive strategies on the other hand are based on reconfiguring the network topology after failure. Basically, an inter-segment topology is formed in order to reestablish connectivity amongst disjoint segments. Reactive strategies can be broken down into two main classes, namely, centralized and distributed, based on the amount of information available during the recovery process. A centralized approach assumes the knowledge of the entire network state, which is exploited to find an optimal recovery solution. The recovery basically boils down to a node placement problem for which there are a number of published heuristics (Cheng et al., 2008; Lloyd and Xue, 2007; Senel and Younis, 2011; Senel and Younis, 2012). Meanwhile, distributed approaches rely on local state information and try to reconnect the network by either having representative nodes from the segments meet at a common point (Joshi and Younis, 2012; Lee and Younis, 2010; Joshi and Younis, 2013) or exploiting the shape of the network topology before failure to determine the recovery paths (Joshi and Younis, 2014; Joshi and Younis, 2015). Although centralized approaches yield optimized solutions, they require

* Corresponding author.
E-mail addresses: yjoshi1@umbc.edu (Y.K. Joshi), younis@umbc.edu (M. Younis).

http://dx.doi.org/10.1016/j.jnca.2016.03.009
1084-8045/

external resources, i.e., aerial support from satellites, aircrafts or UAVs to collect and disseminate global network state information on demand. Such external support may not be available at all times or be feasible due to budgetary constraints. Therefore, distributed approaches are deemed more practical for ad-hoc formed WSNs.

1.1. Contribution

In this paper we study the connectivity restoration problem under resource availability constraints. The problem is motivated by the fact that after a catastrophic failure the surviving segments may not have enough mobile nodes that can serve as relay nodes (RNs) and form a stable inter-segment topology. Basically, most distributed recovery approaches found in the literature, e.g., (Lee and Younis, 2010; Joshi and Younis, 2013; Joshi and Younis, 2014; Joshi and Younis, 2015), are based on the assumption that the surviving segments have sufficient RNs within them and these RNs are available for repositioning without negatively affecting the intra-segment connectivity. This assumption, however, may not hold in practical scenarios where failures randomly take place and a segment may have insufficient RN count to support recovery. To tackle such a challenging recovery problem, we present a novel *R*esource *C*onstrained *R*ecovery (RCR) approach.

RCR aims to reconnect the disjoint segments in a partitioned WSN that has a fixed number 'l' of available RNs that is insufficient for forming a stable inter-segment topology. Therefore, RCR utilizes the available RNs to provide intermittent connectivity amongst the segments. We make the recovery problem even more realistic by restricting the capability of the available RNs. Out of the 'l' RNs available we consider 'l_S' of them to be stationary RNs and 'l_M' to be mobile RNs that can be utilized as mobile data collectors (MDCs). This restriction is due to the fact that some RNs may have low battery life either due to the overhead experienced while participating in the recovery process, or because they may have suffered some damage that impairs their movement. Therefore it is practical for these RNs to be stationary and act as an interface between disjoint segments in order to prolong their lifetime.

Fig. 1 gives a succinct overview of our strategy. Given an ad-hoc WSN, as seen in Fig. 1(a), that suffered a large scale failure due to an external event, the network is partitioned into four disjoint segments as in Fig. 1(b). Our aim is for the disjoint network segments to discover one another and reestablish communication links between them by employing the surviving relay nodes. In the first phase highlighted by Fig. 1(c), segments populate representative RNs towards a common meeting point, the location of which is determined prior to failure and stored within network nodes. If the segment has excess RNs to spare for recovery, they follow the leading RN in a cascaded manner towards the meeting point. Once connected at the center, the RNs exchange information with one another and now know the number of RNs available for recovery and how many of them can be employed as MDCs and stationary RNs. Based on this information exchange, in the second phase the segments are divided into groups whose count is equal to the number of MDCs so that each MDC is assigned a subset of segments to tour. In the segment grouping process, we can utilize some of the stationary RNs as interfaces between various groups to ensure balanced tour loads on the MDCs. In the third and final phase the remaining stationary RNs that are unutilized during the grouping process are employed to shorten the travel path of the largest MDC tours. Fig. 1(d) shows the final reconnected topology.

RCR is validated through simulation experiments. The simulation results show that RCR outperforms competing schemes and produces tours that are not only smaller in total length but they also share the travel load more equitably and thus improve data latency. The rest of the paper is organized as follows. The next section sets RCR apart from existing solutions. Section 3 discusses the system model. Section 4 describes RCR in detail. Section 5 reports the simulation results. Section 6 concludes the paper.

2. Related work

As pointed out in the previous section, proactive strategies that aim to exploit redundancy as a recovery mechanism by forming k-connected topologies do not scale well to handle multi-node failures. In addition, distributed reactive recovery solutions that deal with single node failures e.g., (Das et al., 2007; Akkaya et al., 2010; Senel et al., 2007), cannot be scaled to handle multi-node failures. These solutions require the neighbors of a failed node to collaborate with one another, to either find a replacement for the failed node, or inward movement by all neighbors until connectivity is reestablished. This reliance on neighbors of a failed node does not scale for multi node failures, since in a multi node failure scenario, the scope of failure is unknown, i.e. the nearest healthy node may be many hops away. Surviving nodes will not know in which direction to proceed for recovery unless they store multi-hop information and maintaining a global view of the network state imposes significant messaging and storage overhead.

In the remainder of this section we set RCR apart from published recovery solutions that tackle multi-node failures.

Centralized approaches such as FESTA (Senel and Younis, 2011) and IO-DT (Senel and Younis, 2012) treat recovery as a relay placement problem, which is equivalent to solving for the Steiner Minimum Tree with Minimal Steiner Points (SPs) and Bounded Edge-Length (SMT-MSPBEL) shown to be NP-Hard by Lin and Xue (1999). The solution provides the minimum number and position of RNs that need to be deployed in order to reconnect the network. Although these approaches provide the best possible solution for recovery, they require the entire network state. Hence their use may not be possible in a resource constrained ad-hoc network that does not have access to satellite links or airborne units to provide the entire state. Also these centralized approaches cannot tackle

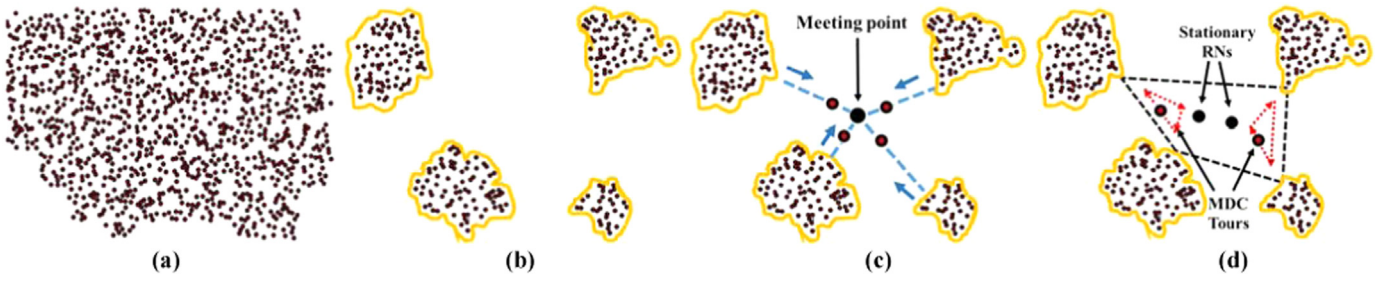

Fig. 1. Overview of RCR. (a) An ad-hoc WSN pre-failure, (b) WSN partitioned into 4 disjoint segments after a catastrophic failure, (c) segments populating RNs towards a common meeting point and discovering one another and (d) two RNs employed as stationary relay nodes to shorten the MDC tours.

recovery when the number of surviving RNs in the damaged WSN is less than the number of SPs.

On the other hand, distributed recovery approaches rely only on the local state, obviously at the expense of optimality. For example, AuR (Joshi and Younis, 2012) utilizes 1-hop information to perform a combination of self-spreading and movement inwards toward the center of the deployment area to guarantee convergence and reconnect the disjoint segments in a partitioned network. Although AuR can handle the resource-constrained recovery problem that RCR aims to tackle, AuR requires all nodes in the WSN to be mobile and thus may not be applicable in all WSN applications wherein a network can be composed of a of a mixture of stationary sensor nodes and mobile relays.

DORMS (Lee and Younis, 2010) and DarDs (Joshi and Younis, 2013) utilize mobile RNs to restore connectivity. The basic idea is to place RNs towards a pre-determined common point in the deployment area along the shortest path and form an inter-segment topology that restores connectivity. Once initial connectivity is restored an optimization phase is run to reduce the number of deployed RNs. SSBR (Joshi and Younis, 2013) and GSR (Joshi and Younis, 2014) pursue a different approach; instead of having a common meeting point, they exploit the pre-failure topology of the network to determine recovery paths that can be populated with RNs in case of failure. While highly effective, they along with DORMS and DarDs operate under the assumption that the disjoint segments collectively have sufficient RNs to form a stable inter-segment topology and do not consider the case where the number of available RNs are insufficient for forming a stable inter-segment topology.

Mobile data collectors (MDCs) or data mules have been used for data collection and dissemination in sparse or fragmented networks wherein direct communication links are absent between different network components (Shah et al., 2003; Jain et al., 2006; Alsalih et al., 2007; Alsalih et al., 2010). The idea of using MDCs as a medium to provide intermittent connectivity presented in these schemes can be utilized in a damaged WSN to restore connectivity amongst disjoint segments when a stable topology cannot be formed due to limited RN availability. Approaches like (Almasaeid and Kamal, 2007; Shen et al., 2005) handle recovery by focusing on data delivery. In (Almasaeid and Kamal, 2007), a data delivery model utilizing MDCs is presented; it basically considers the effect of the number of available MDCs on the end-to-end delay. Meanwhile in (Shen et al., 2005), a MDC tour is designed to minimize data loss rates and looks at the data generation frequency in determining tour paths. However, these approaches are centralized in nature and require the location of segments and scope of failure to be determined by external means. Moreover, they do not consider MDC tour length as a metric.

On the other hand, some MDC-based recovery solutions try to find the shortest MDC tour that covers the segments under each MDC's purview. This reduction in tour not only improves data latency but saves energy, thus extending a MDC's lifetime. Finding the shortest tour though is a traveling salesman problem, which is known to be NP-hard (Applegate et al., 2006) hence the recovery solutions pursue polynomial time heuristics to determine a MDC's tour path. Approaches like MINDS (Joshi and Younis, 2014) and IDM-kMDC (Senel and Younis, 2012) divide the set of disjoint segments into groups; each is toured by a MDC. Both MINDS and IDM-kMDC form a minimum spanning tree (*mst*) over the set of disjoint segments and utilize it as the backbone to divide segments into groups. MINDS (Joshi and Younis, 2014) divides the *mst* around its center and recursively keeps dividing the larger half until there is a group for every available MDC. IDM-kMDC (Senel and Younis, 2012) on the other hand pursues a greedy approach; it initially assigns a MDC to every *mst* edge and then continuously merges the two MDC tours that have the least increase in overall tour length until the number of tours equals the number of available MDCs. However, none of these approaches consider the case where some of the available RNs cannot take part in touring and have to be stationary in the repaired topology.

MiMSI (Abbas and Younis, 2013) deals with a more constrained version of the problem wherein not all available RNs can be utilized as MDCs and some may have to be utilized as stationary RNs. MiMSI uses kmeans++ (Arthur and Vassilvitskii, 2007) to divide disjoint segments into groups that can be toured by a MDC and utilizes stationary RNs to shorten the longest MDC travel path. RCR also deals with recovery under the same constraint as MiMSI. However, unlike other approaches, it not only aims to minimize the total MDC tour length but also focuses on creating balanced MDC tours to ensure that no single MDC is overloaded and thus strives improve data latency.

We compare the performance RCR with MiMSI and show that our approach outperforms MiMSI in all relevant metrics such as total MDC tour length, the maximum touring load on a MDC and the standard deviation amongst all MDC tours.

3. System model

The proposed RCR approach considers an ad-hoc WSN composed of a mix of stationary sensor nodes and mobile relay nodes (RNs) or all RNs that has been partitioned into multiple disjoint segments due to the failure of multiple collocated nodes. We assume that a RN is aware of the size of the deployment area and its position, e.g., using contemporary localization schemes. Prior to failure, the network determines a common meeting point e.g., center of deployment area, which is stored by RNs and is utilized during the recovery process to link up with other survivors. All RNs are assumed to have the same communication range *R*. The surviving nodes at the periphery of failure detect node failures by assessing the number of neighbors they have lost contact with and inability to reach other parts of the network, etc. (Vemulapalli and Akkaya, 2010). The survivors group themselves into a connected segment and act as one entity for recovery.

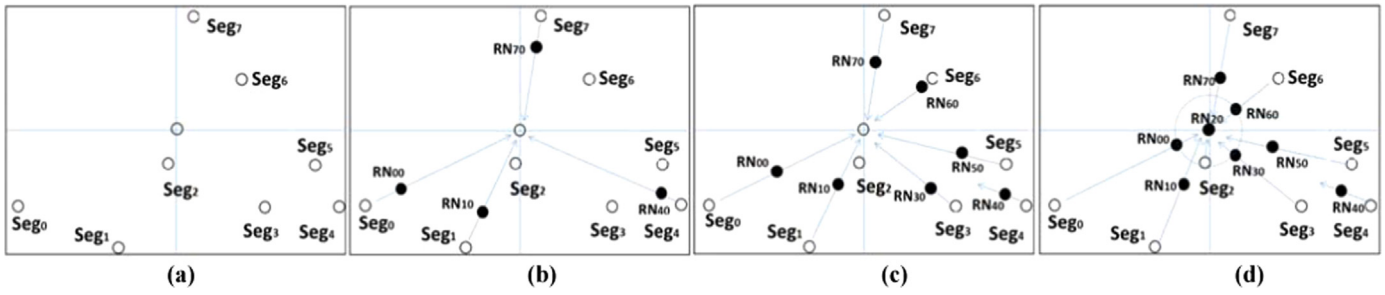

Fig. 2. Initial relay deployment phase. (a) Eight surviving segments in a damaged WSN, (b) RN positions after some time *T*, black circles represent deployed RNs, (c) Position after time $T+\Omega$ and (d) center connected topology achieved.

Each segment picks a RN to lead the recovery process and is followed by other RNs if available. The selection of the leader can simply be based on its proximity to the common meeting point. Fig. 2(a) shows a WSN that has been partitioned into eight disjoint segments Seg_0, Seg_1, …, Seg_7 due to multi-node failure. The center of the deployment area 'O' is considered the common meeting point for the remainder of the discussion. RCR aims to connect these disjoint segments using the surviving RNs.

4. RCR approach

RCR consists of three phases, namely, initial relay deployment, segment grouping phase, and a tour optimization phase. These phases are described in detail in the balance of this section.

4.1. Initial relay deployment

After a network has been partitioned into multiple disjoint segments, the surviving segments do not have any knowledge about each other's position. So to restore initial connectivity and get in contact with other survivors, each segment populates RNs towards a pre-determined common point decided before failure, e.g., the center of the deployment area 'O'. It is assumed that each segment has at least one RN available for recovery purposes, otherwise it will remain unconnected unless it lies in the travel path of another segment's RN. The relocation is led by the leading RN of each segment as shown in Fig. 2(b). To ensure coordination during redeployment and to associate RNs to their respective segments, the relays are uniquely labeled as RN_{ij}, where 'i' is the segment ID which identifies the association of relay and 'j' is the index which distinguishes the order of relays deployed in the path from the segment to 'O'. The leading relay is identified as RN_{i0}. The number of RNs required to reach any node at 'O' from a segment 'i' is:

$$N_{Relays} = \frac{Dist\,(Seg_i, O)}{R} - 1 \tag{1}$$

where $Dist(Seg_i, O)$ is the Euclidean distance between the segment and 'O'.

The farthest segment from the center has to be located on the diagonal of the deployment area and be half the diagonal length away. The maximum time that can be taken by a RN from the farthest segment to reach the center 'O'is given by:

$$T_{Max\_center} = \frac{L_{Diagonal}}{2v} \tag{2}$$

where $L_{Diagonal}$ is the diagonal of the deployment area and 'v' is the average speed of a RN. Therefore for any segment Seg_i located in the interior of the deployment area, the time to reach the center 'O' is given by:

$$T_{to\_center}(i) = \frac{\text{Distance to center}}{v} \tag{3}$$

All segments may not have N_{Relays} available RNs. Therefore to ensure proper utilization of the deployed RNs, each segment Seg_i waits for $T_{Wait}(i)$ amount of time for leading RNs of segments farthest from 'O' to come with contact range of the nodes of Seg_i, before Seg_i decides to sends RNs towards 'O', where:

$$T_{wait}(i) = T_{Max\_center} - T_{to\_center}(i) \tag{4}$$

This wait time ensures that an interior segment will not unnecessarily deploy RNs if it lies on the path of outer segments towards 'O', and thus saving the energy of its RNs. RCR aims to minimize the number of RNs deployed to reach the center but also reduce the RN travel distance; therefore T_{Wait} allows segments to discover one another before reaching 'O'. The inward motion allows multiple leading RNs to get in range of one another on route to the center, where they exchange their respective segment information and the closest RN to the center amongst them continues until reaching 'O' while the others stop.

For example in Fig. 2(a), each segment has just one mobile RN available for recovery. After a failure, Seg_0, Seg_1, Seg_4, Seg_7 deploy their only RN towards 'O' before other segments, as seen in Fig. 2 (b). This is due to the fact that they lie on the periphery of the WSN and hence have a smaller T_{wait} time compared to interior segments. As seen in Fig. 2(b), during the inward motion towards the center, RN_{40} from Seg_4 comes within the communication range of nodes in Seg_5 and stops. After Seg_5's T_{wait} is over, its leading RN_{50} is deployed towards the center as seen in Fig. 2(c); such a leading RN also has information regarding the position of RN_{40} and Seg_4 that will be utilized in later phases. As seen in Fig. 2(d), the leading RNs of Seg_6 and Seg_7, Seg_3 and Seg_5 come in the communication range of one another and the closest RN to 'O' obtains the segment information from the other RNs, and continue toward the center. Similarly RN_{10} of Seg_1 comes in contact with Seg_2 and stops. Given the close proximity of Seg_2 to 'O', it has the largest T_{wait} and is the last to be deployed and proceeds to reach 'O'. RN_{00}, RN_{30} and RN_{60} of the other segments stop when they come within its communication range of RN_{20}, resulting in a center connected topology as seen in Fig. 2(d).

The maximum RN count required to establish a connected inter-segment topology will be less than or equal to the sum of RN_O and all RNs on the paths from the individual segments toward 'O' until becoming in range of RN_O. Thus, the maximum number of RNs that can be deployed during initial relay deployment can be given by:

$$Max_{Relays} \leq \sum_{i=1}^{\#segments} N_{Relays} + 1 \tag{5}$$

Note that in practice the actual RN count is often less than Max_{Relays} since some of the RN paths from the individual segments get merged during the inward relay deployment. Since we are considering a resource constrained WSN, the number of RNs available for connectivity will always be less than Max_{Relays}. For example in the WSN depicted in Fig. 2 the number of relays needed to establish a stable inter-segment topology without taking merging into account is 20, although it only has 8 RNs available for recovery.

Theorem 1. *RCR converges to an initial connected topology in constant time.*

Proof. In the initial phase, RNs are populated by all surviving segments until a leading RN becomes reachable to the center, i.e., becomes R units away from 'O', or merges with a segment while on route to 'O'. Only one of these leading RNs will eventually be positioned at 'O'. In the worst case scenario the surviving segments can be located at the corners of a rectangular deployment area, so the maximum distance between a segment and the center will be half the diagonal length. Assuming at least one segment is located at the corners of the deployment area and travel inwards at a speed 'v', the time taken will be given by distance divided by speed, i.e., $\left(\frac{L_{Diagonal}/2}{v}\right)$, which is a constant. □

Lemma 1. *There can only be a maximum of six segments that do not merge before reaching the center.*

Proof. For segment representatives to meet at the center, they need to be separated by 'R' when they come in contact with RN_O. As shown in Fig. 3, only six such positions can exist since if a pair of adjacent segments are separated by 'R' and equidistant from the center, they form an equilateral triangle which has a center angle of 60°. The center subtends an angle of 360°; therefore there exist only 6 segments that will meet while being 'R' away from the

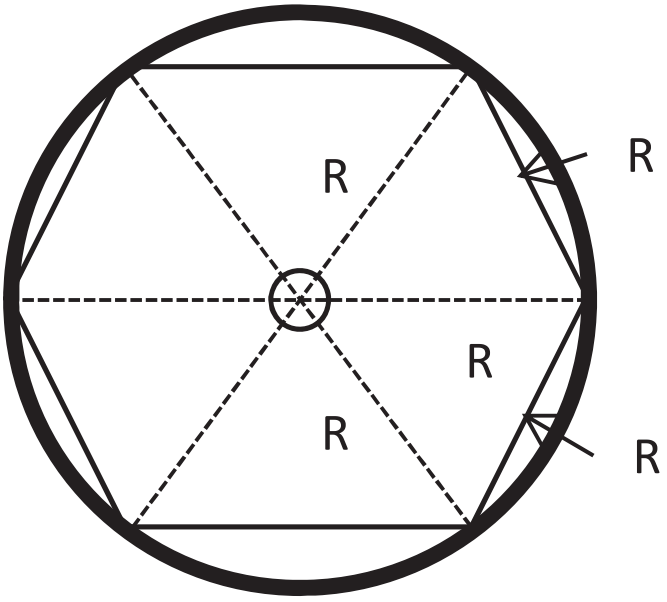

Fig. 3. The topology at the center with segments separated from each other and the center by *R*.

center. Thus, for $N_{Seg} > 6$, segments will merge before reaching the center and the number of employed RNs and the total distance they have to travel in the initial phase will be reduced significantly.

4.2. Segment grouping

After the initial relay deployment phase, the leading RNs of different segments are connected with one another around '*O*'. RN_O would then know the position of all disjoint segments and the resources available within them for recovery. RN_O divides the available RNs into l_S stationary RNs and l_M RNs to be utilized as MDCs based on a pre-determined user criterion that could vary according to the application and deployment scenario and could also factor in the remaining energy level of the RN, or the relay's condition, i.e., physical damage that may constrain movement. In order to establish inter-segment connectivity, RCR forms l_M groups of segments; the segments of each group is to be toured by a MDC. The objectives for segment grouping are: (i) the tour length of individual MDCs is to be balanced in order to reduce the maximum data delivery delay and equalize the motion overhead, (ii) simplify the inter-segment data forwarding by relaying data among MDCs through segments, and (iii) minimize the travel overhead on individual MDCs.

To achieve the above objectives, RCR first picks a terminal, i.e., one of its nodes, S_i for each segment Seg_i to act the segment's interface with the MDC for date exchange. Thus, the aim of segment grouping becomes to identify terminals that can serve as rendezvous points between l_M groups such that MDC tour length for each group is nearly equal. This ensures that the load on the MDCs is balanced and reduces the data delivery latency. Popular proximity-based clustering approaches like kmeans++ focus on minimizing the distance between the group members, i.e., minimize the intra-cluster tour, and do not factor in inter-cluster connectivity as a design goal. In other words, after clusters are formed, these approaches identify terminals that can serve as gateways for data exchange between the clusters, which often results in poor choice of rendezvous points. Unlike these proximity-based clustering approaches, RCR determines the gateway terminals around which clusters can be built such that the clusters formed have balanced tours.

To identify the gateway terminal between segments groups RCR employs the concept of eccentricity of a graph (Wuchty and Stadler, 2003). Given a graph $G(V,E)$ that consists of the set of vertices V and the set of edges E, the eccentricity of a vertex $v \in V$ is defined as the distance from v to the most distant vertex. A vertex of minimum eccentricity is called the center of a graph. By modeling the terminals as a graph, RCR opts to employ the center of such a graph as a rendezvous terminal in order to minimize the maximum data delivery delay, i.e., achieving objective (i) above. Having a rendezvous terminal that is part of the tour of two MDCs obviously simplifies data exchange between them and thus achieves objective (ii). RCR pursues two methods for factoring in the distance $d(x,y)$ between two vertices x and y, namely, (1) $d(x,y)$ is the length of the shortest multi-hop path between x and y, and (2) $d(x,y)$ is the length of the edge between x and y. The former is intuitive since all terminals that are to be connected by an MDC, which will have to hop amongst them in its tour. Since RCR opts to minimize the travel overhead, i.e., objective (iii) above, we consider the minimum spanning tree (*mst*) of the terminals as the underlying terminal graph in this case. The cost of constructing an *mst* is $O(|E| \log|V|)$. In summary, RCR considers the following two scenarios:

1. What is the optimal rendezvous terminal if we consider only *mst* edges, i.e., if a path connecting x and y does not exist we set $d(x,y)=\infty$.
2. What is the optimal rendezvous terminal if we assume there exists an edge connecting every pair of vertices.

These two scenarios can be recast as optimization problems based on the distance matrix $D=(d(x,y))$ of G. The eccentricity $e(x)$ of a vertex x in G and the radius of a graph $\rho(G)$ is defined as:

$$e(x) = \max_{y \in V} d(x,y) \tag{6}$$

$$\rho(G) = \min_{x \in V} e(x) \tag{7}$$

The center of G is the set of vertices that have the minimum eccentricity, i.e.,

$$C(G) = \{x \in V | e(x) = \rho(G)\} \tag{8}$$

So we solve for the graph center in both scenarios; one where we consider only *mst* edges in $D_1=(d(x,y))$, and in the other we consider an edge existing between every pair of vertices $D_2=(d(x,y))$. We find candidates for the graph center in both these scenarios and qualify the best choice based on the tours that could be formed when a candidate vertex is picked as a rendezvous terminal. Basically, we utilize the *mst* as the underlying structure to divide the set of terminals around the candidate vertex (Joshi and Younis, 2014) and determine the tour lengths T_1 and T_2 in each case. The candidate vertex which results in the least difference in tour lengths (T_1 and T_2) is chosen as the rendezvous terminal.

The question that arises is why RCR considers two scenarios to select the best rendezvous terminal. The reason is because of the vagaries of the *mst*, which is highly dependent on the segment positions; sometimes just considering the *mst* edges in the distance matrix $D=(d(x,y))$ of G results in a non-ideal choice. This is illustrated in Fig. 4(a), which shows four terminals S_0–S_3 representing their respective segments Seg_0–Seg_3. For these segments the rendezvous terminal is to be determined. As seen in Fig. 4(b), the *mst* edges connecting the terminals are of length a–c, respectively with $(b<c<a)$. When using only *mst* edges in $D_1=(d(x,y))$, we end up with S_2 as the graph center since $(b+c)<(a+b)$ and thus S_2 has the minimum eccentricity, i.e., $d(S_2, S_0) < d(S_1, S_3)$. Meanwhile in the second scenario depicted in Fig. 4(c) and (d), we assume that an edge exists between every pair of vertices, which

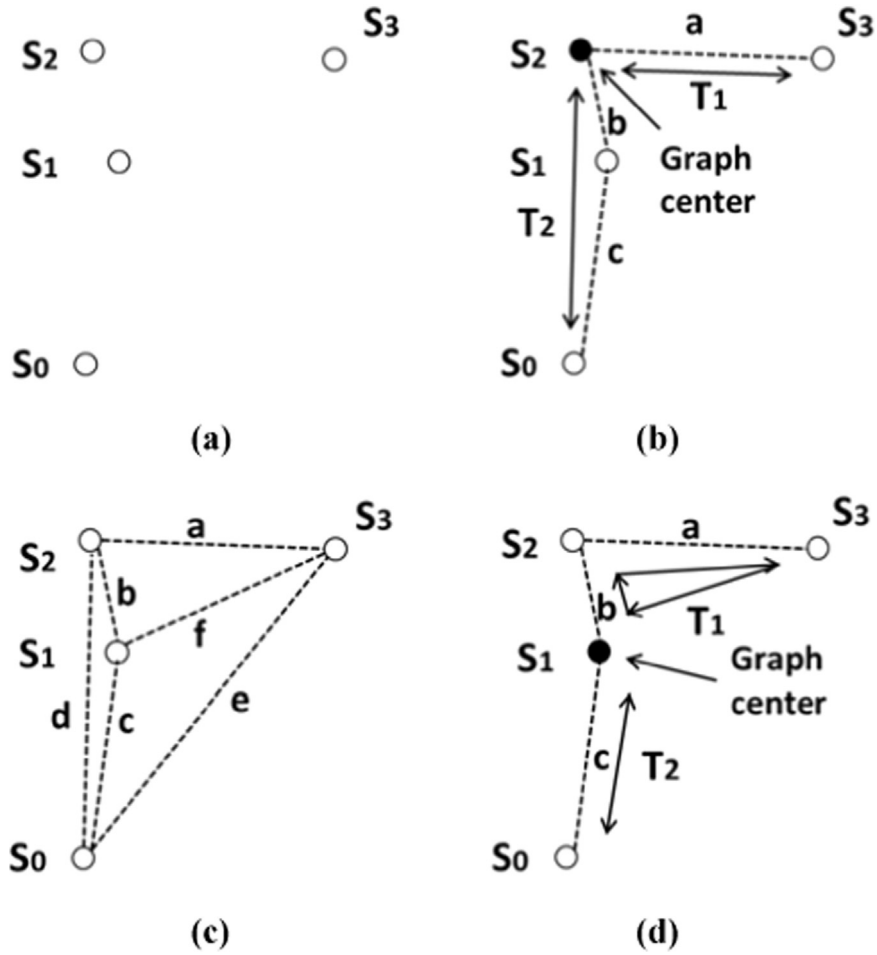

Fig. 4. Illustrating the selection of graph centers for segment grouping; (a) an example topology with four terminals S_0–S_3 representing their respective segments Seg_0–Seg_3; (b) scenario #1 where only *mst* edges are considered in the distance matrix, results in identifying S_2 as the graph center; (c) scenario #2 where we consider an edge to exist between every pair of vertices; and (d) the graph center for scenario #2 is located at S_1.

leads to identifying S_1 as the graph center. The resulting tours show that not only is the tour length more equitable if S_1 is used as a rendezvous terminal for this topology, but the overall tour length is also shorter. Thus, by choosing the graph center that results in the most balanced division, we obtain two nearly equal sized groups.

The way RCR performs segment grouping differs based on the availability of l_S RNs as follows:

(1) $l_S=0$: If all RNs can be utilized as MDCs, we simply determine the graph center and divide the *mst* around it. We recursively keep dividing the larger of the resulting parts, i.e., find the graph center of that part, until we obtain l_M groups.
(2) $l_S>0$: If some of the RNs have be stationary, RCR utilizes them to find a graph center around which a more equitable group sizes can be obtained. The question is where to place the stationary RNs so as to improve segment grouping. To answer this question, RCR models the recovery problem as RN placement problem which is equivalent to finding the Steiner Minimum Tree with Minimal Steiner Points and Bounded Edge-Length (SMT-MSPBEL). SMT-MSPBEL is an NP-hard problem for which numerous heuristics have been proposed. Among published solutions, RCR employs IO-DT (Senel and Younis, 2012) which is one of the latest and most effective heuristics. The Steiner points (SPs) determined by IO-DT help in pointing out potential positions where stationary RNs can be deployed to act as gateways between segments. Unlike the case of $l_s=0$, RCR considers the SPs, identified by IO-DT, along with the segment terminals as vertices in G. We determine the graph center and if it lies on a SP we mark it as a place to deploy a stationary RN and decrement the l_S count. This process continues until we obtain l_M groups. In case if we run out of stationary RNs before obtaining l_M groups, the unused SPs are removed from G and RCR proceeds as in the case of $l_S=0$, above. As will be shown in the Section V, our approach yields finer groupings because by having more vertices in the *mst*, we have shorter edges in G. The grouping process will be illustrated through a detailed example later in this section.

Lemma 2. *If $l_S>0$, RCR forms an SMT-MSPBEL utilizing the IO-DT heuristic whose time complexity is $O(n^2)$ (Senel and Younis, 2012).*

Lemma 3. *In an mst, the graph center can be determined in $O(n)$.*

Proof. The longest path in an *mst* can be found by running breadth-first search (BFS) rooted at any vertex say v_i in the *mst* and finding the farthest vertex v_k, and then doing the same for v_k to determine the farthest vertex from it in the *mst*. The time complexity of these steps is $O(E)$ or $O(n-1)$ as there are $n-1$ *mst* edges over n vertices. The graph center is located at the vertex whose distance to the farthest vertices of the longest path is the smallest.

The longest path can comprise of at most $n-2$ vertices and $n-3$ *mst* edges. Thus, finding the center of the *mst* is $O(n)$.

Lemma 4. *In a graph G where an edge exists between every pair of vertices the graph center can be determined in* $O(n^2)$.

Proof. In an *mst* comprising of n vertices, in order to find the longest edge each vertex needs to check its distance to the remaining $(n-1)$ vertices, doing this for every vertex leads to a time complexity of $O(n^2)$.

Theorem 2. *The runtime complexity of determining all graph centers in the segment grouping phase is* $O((l_M-1)n^2)$.

Proof. Most of the work in this phase is done in recursively splitting the segments until we obtain l_M segment groups. This takes at most (l_M-1) iterations. For each iteration it takes $O(n)$ and $O(n^2)$ time to find the graph center, according to Lemmas 3 and 4 above. Therefore, the runtime complexity of one iteration is $O(n^2)$ and for (l_M-1) iterations the overall time complexity will be $O((l_M-1)n^2)$. □

4.3. Tour formation

At the end of the segment grouping phase, we obtain l_M groups each with its own assigned MDC. In this subsection we explain how the MDC will tour the segments in its assigned group. Each segment Seg_i selects a terminal S_i to act as the segment's interface with the MDC for date exchange. An MDC tour T_S is defined as the shortest possible cycle that visits every terminal $S_0, S_1, S_2, S_{3....} S_q$ in its list for data collection and dissemination and returns to its starting position. Finding the shortest possible MDC tour is equivalent to the Euclidean Traveling Salesman Problem and is NP-Hard (Applegate et al., 2006). Also for data exchange to take place between a MDC and a terminal, the MDC must pass within the transmission range of the terminal, we call these data exchange stops as collections points. This section describes how RCR forms the smallest MDC tour and determines the collection points for the MDC. The tour formation can be divided into two cases based on the number of terminals that need to be covered by the MDC.

- *No. of terminals* $=2$: This is the simplest case to handle, here an MDC is used to provide connectivity between two terminals. The shortest path between two points is a straight line connecting them. If the distance 'δ' between two points is $R<\delta \leq 2R$, where R is the communication range, the MDC will be stationary and positioned at the midpoint of the line connecting the two terminals, as seen in Fig. 5(a). If $\delta > 2R$, the collection points for data exchange are located at the intersection of the transmission discs of terminals S_1 and S_2 of radius R and the line connecting them. As seen in Fig. 5(b), the MDC will move back and forth between collection points P_1 and P_2 for data exchange.
- *No of Terminals* >2: In case a MDC has to tour more than two terminals, RCR forms a simple polygon where all terminals are located on its boundary. To do so RCR first constructs a convex hull over the terminals and then includes interior terminals by replacing their closest edges on the convex hull. To illustrate, we consider the example in Fig. 6(a), where seven terminals are to be toured by an MDC. As seen in Fig. 6(b), we determine the convex hull, resulting in 5 convex and 2 interior terminals. For each interior terminal, RCR finds the closest convex edge as shown in Fig. 6(c) and then that edge is adjusted to include the interior terminal as well, resulting in a simple polygon depicted in Fig. 6(d).

The simple polygon obtained gives us a MDC tour that covers all terminals, but we still need to determine the collection point for every terminal that an MDC need to stops at. For the shortest

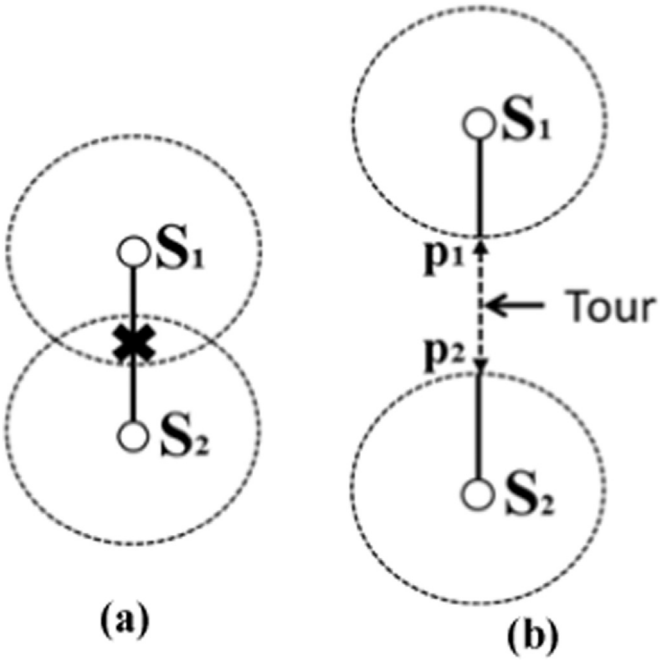

Fig. 5. Determining the tour between two terminals when the distance between them is, (a) $R<\delta\leq 2R$ and (b) $\delta>2R$.

MDC tour, the collection points have to lie in the interior of the polygon. The following theorem provides a guideline for selecting the collection point for a terminal.

Theorem 3. *The maximum reduction in the perimeter of a simple polygon of terminals is achieved when the collection point is located along the angle bisector.*

Proof. Consider Fig. 7, which shows a part of the polygon with three terminals. Assume that our aim is to determine the collection point P_1 for terminal S_1, which will serve as a tour stop for a MDC. We know that P_1 has to be located in the interior of S_1, and can be at a maximum distance R, where R is the communication range of a RN. Let the collection point P_1 lie along a line that divides the angle θ made by S_2-S_1-S_3 into θ_1 and θ_2, respectively. Since for the initial tour determination we consider only those terminals of a simple polygon, i.e., the vertices of the convex hull determined in Fig. 6(b), we know that $\theta<\pi$ therefore $\theta_1<\pi$ and $\theta_2<\pi$. As seen in Fig. 7, the reduction in perimeter or tour length η caused by having a collection point at P_1 can be approximated by:

$$\eta = d_1+d_2, \quad \text{where } d_1 = R\cos\theta_1 \text{ and } d_2 = R\cos\theta_2$$

Substituting in d_1 and d_2 we get

$$\eta = R\cos\theta_1 + R\cos\theta_2 = R(\cos\theta_1 + \cos\theta_2)$$

Therefore for the reduction in perimeter to be maximal we need to maximize the value of $(\cos\theta_1+\cos\theta_2)$, which is equivalent to minimizing θ_1 and θ_2. Since $\theta=\theta_1+\theta_2$ we can replace $\theta_2=\theta-\theta_1$ and η becomes:

$$\eta = R(\cos\theta_1 + \cos(\theta-\theta_1))$$

To maximize η we differentiate both sides with respect to θ_1.

$$\frac{d\eta}{d\theta_1} = R((-1)\sin\theta_1 + (-1)(-1)\sin(\theta-\theta_1))$$

The maximum θ_1 corresponds to $\frac{d\eta}{d\theta_1}=0$, thus

$$R(-\sin\theta_1 + \sin(\theta-\theta_1)) = 0$$
$$\Rightarrow \sin\theta_1 = \sin(\theta-\theta_1)$$
$$\Rightarrow \theta_1 = \theta-\theta_1$$

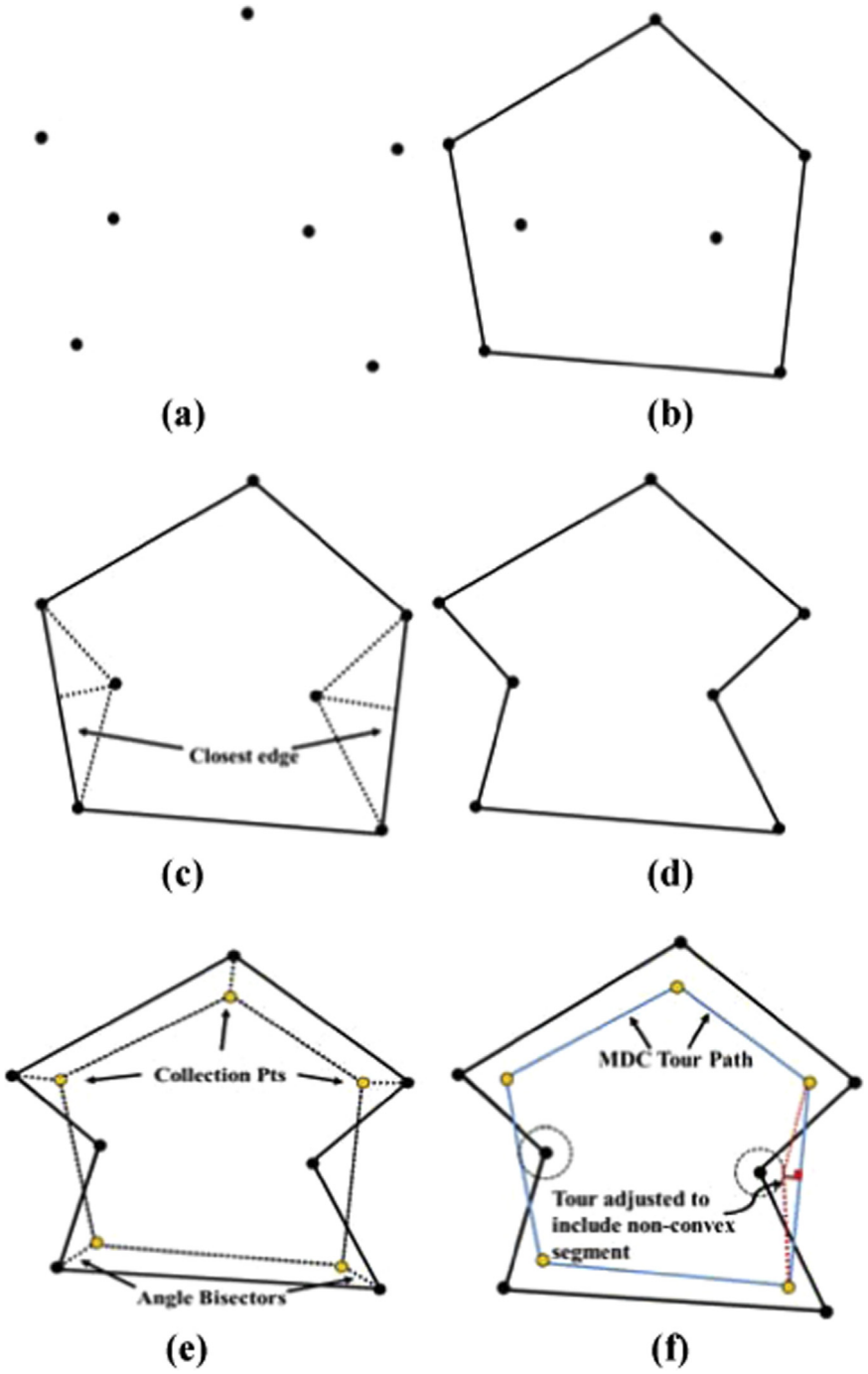

Fig. 6. Illustrating the tour formation steps for a group of more than 2 terminals; (a) the layout of the terminals that need to be toured by an MDC; (b) the convex hull over the terminals, (c) interior terminals find their nearest convex edge, (d) a simple polygon is formed by accounting for interior points, (e) collection points are determined along angle bisectors for the convex vertices of the simple polygon forming a convex tour, and (f) the tour is adjusted to account for collection points for any interior terminals.

Thus, $\theta_1 = \frac{\theta}{2}$ and substituting in $\theta_2 = \theta - \theta_1$, $\theta_2 = \frac{\theta}{2}$. □

Based on Theorem 3, having the collection point located on the angle bisector results in the maximum reduction in length, hence resulting in a smaller tour. So we determine the collection points for all terminals in the following manner. First we consider all the convex terminals in the simple polygon that we found in Fig. 6 (b) and determine their collection points by locating a point R away along their angle bisectors towards the interior of the polygon. Joining these collection points gives us our initial convex tour which is nothing but the perimeter of the convex polygon as shown in Fig. 6(e). If there are any interior terminals, we simply check their nearest convex tour edge and if it is within the transmission range, then the tour does not get adjusted since the MDC will pass in range of the said interior terminal. In case the terminal is more than R units away from its nearest convex edge, the collection point is computed by: (i) finding the point of intersection of the perpendicular line from the terminal to the closest tour edge and its transmission disc, and then (ii) the tour edge is adjusted accordingly to account for the new collection point. This case is illustrated in Fig. 6(f).

Fig. 8 illustrates the tour formation in competing approaches such as MINDS (Joshi and Younis, 2014), IDM-kMDC (Senel and

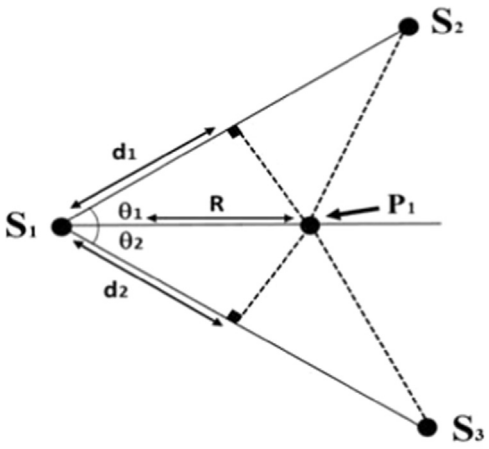

Fig. 7. Determining the collection point of a MDC.

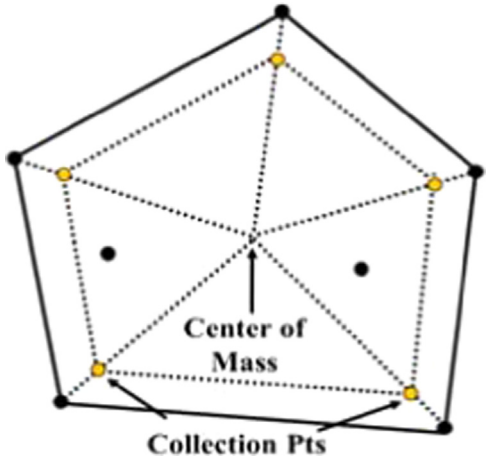

Fig. 8. Illustrating the tour formation steps for competing approaches i.e. IDM-kMDC, MiMSI, MINDS. Basically, the collection points are determined along the line connecting the terminals to the center of mass.

Younis, 2012), and MiMSI (Abbas and Younis, 2013). They determine the tour by forming a convex hull of the terminals and then determine the center of mass. The point of intersection of the transmission disc of a terminal with a line connecting it to the center of mass serves as a collection point, i.e., the collection point is at a distance R on the line between a terminal and center of mass as seen in figure. A valid question would be why RCR ignores the center of mass and instead uses angle bisectors to determine the collection points. One reason is that by considering the convex hull, we ignore the interior terminals, and thus consider a much larger area than actually occupied by the terminals. Therefore the center of mass may conceivably be in an unfavorable position resulting in poor choices of collection points. Secondly Theorem 3 implies that populating along angle bisectors results in the largest reduction in perimeter. One can also ask why the collection points are not determined using the initial convex hull found in Fig. 6 (b) since the convex vertices of the simple polygon are utilized in determining the initial tour stops. The reason for obtaining the simple polygon is to minimize the angle θ at the convex vertices, thereby yielding further reduction in perimeter as compared to just using the convex hull and also helps in the final tour optimization phase, as will be explained in the next subsection.

Theorem 4. *The runtime complexity of RCR tour formation for a set of n terminals is $O(n^2)$.*

Proof. The convex hull over a set of n points can be formed in $O(n\log n)$ using Graham scan. To form a simple polygon, each interior point needs to find its nearest convex edge. Say there are 'k' interior points so each point needs to check $(n-k)$ edges to determine the closest one, resulting in a time complexity of $O(k(n-k))$ or $O(n^2)$. The collection point can be determined in constant time for each vertex of a simple polygon giving a total time complexity of $O(n)$. So the overall time complexity of tour formation is $O(n^2)$. □

4.4. Tour optimization

In this final phase, RCR aims to use any of the l_S RNs that were unutilized in the segment grouping phase in order to further reduce the length of the MDC tours. A sorted queue in descending order of the MDC tour lengths is maintained and the available stationary RNs are then deployed to reduce the tour length of largest tour in the queue until all stationary RNs are utilized. Fig. 9 is used in this subsection to illustrate the tour optimization for a MDC tour that spans seven terminals and has $l_S=3$. The objective of tour optimization is to equalize the travel load on the MDCs. Two main optimization steps are applies.

Optimization #1: Generally, there are multiple positions where the RNs can be deployed. One option is to place a RN on a polygon edge between two terminals in order to shorten the distance between them; however such placement has no impact on the tour length since those terminals need to be visited regardless of the added RN. On the other hand, Theorem 3 implies that placing RNs along the angle bisectors results in the best choice to reduce the perimeter, so we can apply the same logic to reduce a MDC tour length.

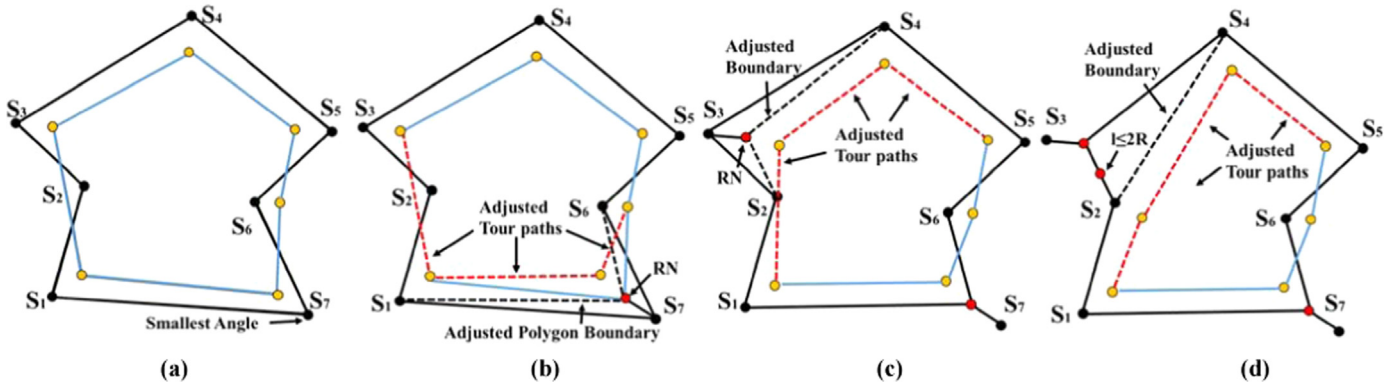

Fig. 9. Tour optimization of a MDC tour that spans seven terminals and $l_S=3$. (a) The smallest angle located at S_7, (b) RN placed 'R' away along the angle bisector of S_7, (c) next smallest angle located at S_3, and (d) tour optimization when a polygon edge has a length $\delta \le 2R$.

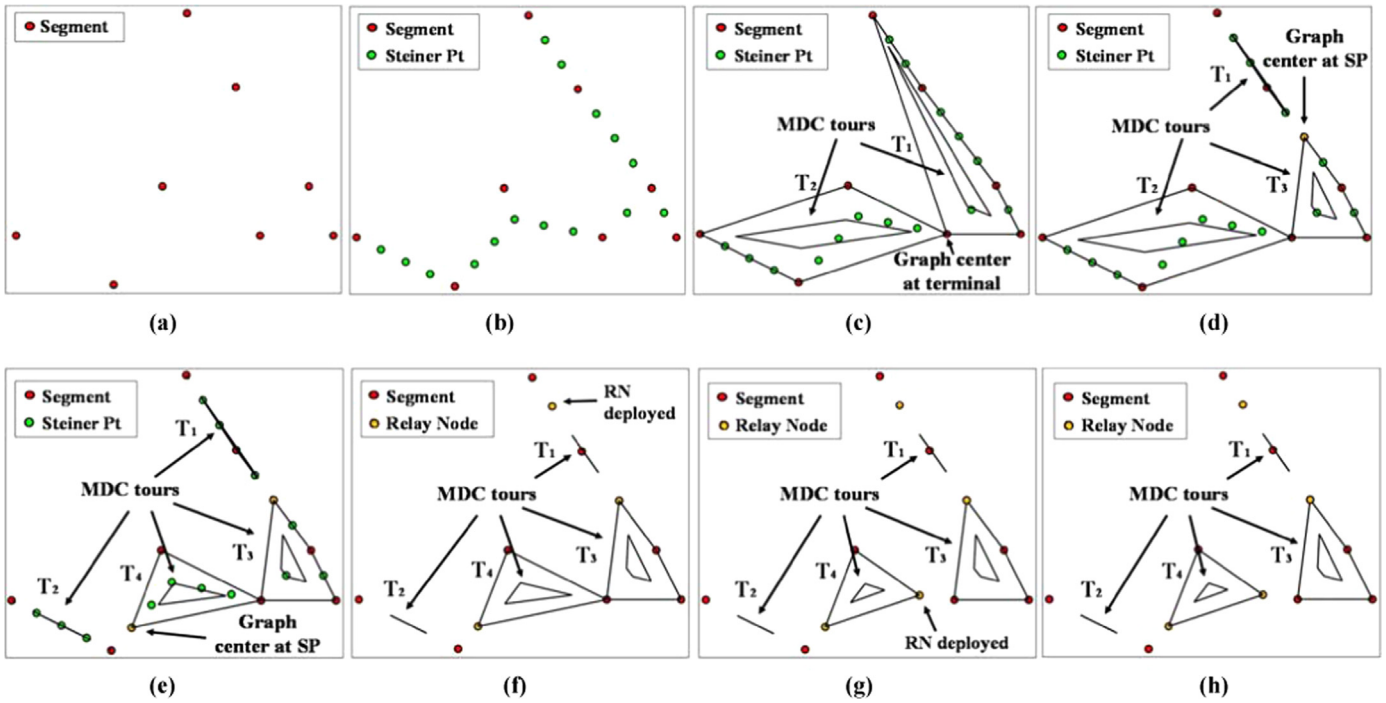

Fig. 10. An illustrative example that demonstrates how RCR carries out segment grouping, tour formation and optimization when $l_M=4$ and $l_S=4$. (a) Initial set of eight terminals to be connected. (b) Since $l_s > 1$, find Steiner points and determine the graph center. (c) Divide mst into two groups around the graph center forming tours T_1 and T_2. (d) Partition the larger T_1, the graph center located at SP, so deploy a RN. (e) Divide the larger T_2, the graph center located at SP, deploy a RN. (f) Number of Tours is equal to l_M, number of available MDCs and $l_S=2$. Tour optimization for largest tour T_1 by deploying a RN. (g) Largest tour T_4 optimized by placing a RN along the angle bisector of the smallest angle. (h) Final reconnected topology with 4 MDC tours.

Theorem 5. *A reduction of approximately $2R\cos\frac{\theta}{2}$, in the perimeter of a polygon could be achieved, when an RN is placed along the angle bisector of the vertex (terminal) that has the smallest angle in the simple polygon.*

Proof. Based on Theorem 3 the maximum reduction is achieved if a RN is placed at a distance R from the terminal along the angle bisector. Using the equation of the perimeter in (1)

$$\eta \approx R(\cos\theta_1 + \cos(\theta-\theta_1))$$
$$\approx R\left(\cos\frac{\theta}{2}+\cos\frac{\theta}{2}\right)$$
$$\approx 2R\cos\frac{\theta}{2}$$

Therefore by having the smallest θ we can obtain the largest reduction which is nothing but choosing the smallest angle in the simple polygon. □

As shown in Fig. 9(b), the smallest angle is located at terminal S_7 and a RN is placed at its collection point to shorten the MDC tour. In the next iteration the smallest angle is at terminal S_3 as shown in Fig. 9(c).

Optimization #2: As argued above, placing an RN between two terminals connected by a polygon edge does not affect the overall MDC tour length; however, if we have a polygon edge that connects a convex terminal with an interior terminal of length $\delta \le 2R$, then by placing a RN on such an edge we can connect the 2 terminals. In other words, we can shrink the polygon by removing the convex terminal from consideration, and modifying its other incident edge to connect to the interior terminal. This leads to a smaller simple polygon and which in turn reduces the MDC tour. Fig. 9(d) illustrates the idea through an example. In Fig. 9(d), the length of the edge $S_2S_3 \le 2R$, if we place an RN at the midpoint of S_2S_3, terminals S_2 and S_3 will be connected with one another and we can replace the edges S_3S_4 and S_2S_3 with the edge S_2S_4; thus we obtain a smaller simple polygon.

Theorem 6. *If there exists a polygon edge with length $\delta \le 2R$, that connects a convex terminal and an interior terminal, by placing a RN on that edge, a maximum perimeter reduction of 2R can be achieved.*

Proof. Let us again consider Fig. 9(d), particularly the edge S_2S_3. The reduction in the perimeter η when applying the optimization to S_2S_3 is given by:

$$\eta = S_2S_3 + S_3S_4 - S_2S_4$$

Considering the triangle $S_2S_3S_4$, and applying the triangle inequality;

$$S_2S_4 + S_2S_3 > S_3S_4,$$

Subtracting S_2S_4 from both sides we get,

$$S_2S_3 > S_3S_4 - S_2S_4,$$

Note that S_2S_4 will always be longer than S_3S_4, since the interior terminal S_2 has to lie on the right of S_3 since RCR chooses the closest convex edge to S_3 when forming the simple polygon. Hence,

$\eta = S_2S_3 - \Delta$, where Δ is a positive number

Thus, we can conclude that $\eta \le S_2S_3 \le 2R$. □

The removal of the edge S_2S_3 in Fig. 9(d) reduces the size of the simple polygon and thus yields a reduction in the MDC tour length as well. The removal of S_2S_3 also reduces the angle subtended at S_4 and further shrinks the tour length when the collection point is calculated. Therefore, by forming a simple polygon RCR simplifies the identification of edges $\le 2R$ in length, as only edges that are added when accounting for interior terminals are considered in this step.

In summary, RCR carries the tour optimization based on the following criteria; (1) in the simple polygon, find the terminal subtending the smallest angle and locate a point R away on the angle bisector where a RN can be placed; (2) if there exists an edge with a convex and non-convex terminal as its endpoints and has an edge length $\le 2R$, place a RN at the midpoint of the edge and

```
n← No of disjoint segments after failure
l_S← Number of RNs
l_M← Number of MDCs
R← Relay Communication Range

Initial Relay Deployment Phase [For every Seg_i]
 -Identify Leading RN RN_i0 for Seg_i and other spare RNs if
available to follow in cascaded manner.
 - Set T_Wait and wait until T_Wait=0.
DO
   IF disttoCenter>R
    Move RN towards Center O
    IF meets RN_jk or Seg_j where j≠i  THEN
       Exchange Seg and RN information and STOP
    ENDIF
   ELSE
    Become center RN and STOP
   ENDIF
 WHILE (RN_i0≠STOP)

Segment Grouping [l_S, l_M, n ]
 -IF l_S> 1
   Determine SPs from SMT-MSPBEL over 'n'
   n= No of  SPs+n
 -ENDIF
 -Mst(V, E) ← Minimum Spanning Tree over n
 - Insert in Tours(T_1(V));
 - NTours=1;
 - WHILE NTours< l_M
    T_i← Remove largest from Tours();
    C_i← FindGraphCenter(T_i);
    IF C_i==SP
      l_S--;
      IF l_S==0
       Remove SPs from Tours();
      ENDIF
   T_j, T_k ← PartitionTour(T_i, C_i);
   FindTourCost(T_j);
   FindTourCost(T_k);
   Delete T_i from Tours();
   Insert in Tours(T_j);
   Insert in Tours(T_k);
   NTours++;
 -END WHILE

Form Tours [Tours[]]
 -Form Convex Hull for all terminals in T_i
 -Find Nearest edge to Interior pts and form Simple
Polygon.
 -Calculate Collection points for all convex pts in T_i along
    angle bisectors, forming initial tour.
 -Determine collection points for non-convex terminals and
    adjust initial tour if necessary.

Tour Optimization [ Tour ]
WHILE l_S>0
   Along largest tour deploy RN either along smallest angle
   or if edge length (Convex to non-convex terminal) ≤ 2R
   and deploy along which ever gives maximum savings.
```

Fig. 11. Pseudocode for RCR Algorithm.

remove the edge from the polygon. If both (1) and (2) are applicable at the same time RCR chooses the option that yields the most reduction in tour length for placing a RN.

Theorem 7. *The time complexity of tour optimization is* $O(l_S n \log n)$, *where n is the number of terminals.*

Proof. The smallest angle in the simple polygon can be found by only checking the 'n' convex vertices and sorting them in $O(n \log n)$. Finding edges of length $\leq 2R$ between convex to non-convex vertices can be done in $O(k)$ time, where k is the number of interior terminals. Since the addition of a RN affects only the edges incident on the vertex or edge being replaced, we do not need to recalculate the entire tour but simply replace the affected edges with new ones which can be done in constant time. The number of iterations for the tour optimization depends on the number of stationary RNs available which can at most be l_S. Therefore the time complexity of tour optimization is $O(l_S n \log n)$.□

4.5. Illustrative example

Fig. 10 illustrates the application of RCR after the initial relay deployment phase is complete. Fig. 10(a) shows eight terminals that need to be connected by utilizing 4 MDCs and 4 stationary RNs. RCR divides the segments into 4 groups one for each MDC. Since $l_S > 1$, we form the SMT-MSPBEL utilizing IO-DT and incorporate the SPs found with the terminals in Fig. 10(b).

We find the graph center over the set of terminals and SPs and split it into two tours T_1 and T_2 and keep recursively splitting the larger group until we obtain 4 groups, one for each MDC. In case the center falls on a SP we mark it as a location to populate a RN and decrement l_S as shown in Figs. 10(d) and 9(e). Once we have obtained the requisite number of groups we try to optimize the tours if we have any stationary RNs left. In this example we utilized two RNs during segment grouping and thus have two RNs available for tour optimization. The tours are maintained in a queue of descending order of their length, with the largest tour first. As indicated in Fig. 10(e), T_1 is the largest tour and we employ a RN to shorten its length as seen in Fig. 10(f). T_1 is reinserted it back in the tour queue. Now T_4 is the largest tour, RCR deploys the RN along the angle bisector of the smallest angle, as seen in Fig. 10 (g). Since there no more RNs left to deploy, RCR terminates. A pseudo code summary of RCR is provided in Fig. 11.

4.6. Overall runtime complexity

Theorem 8. *The time complexity of RCR is* $O((l_M - 1)n^2)$.

Proof. In RCR, the grouping process runs at most $l_M - 1$ times to obtain l_M groups. The time to obtain the graph center is $O(n^2)$ by Lemmas 3 and 4. The tour formation also takes at most $O(n^2)$ as seen in Theorem 4 giving an overall time complexity of is $O((k-1)n^2)$. The tour optimization phase runs in $O(l_S n \log n)$. Hence the overall time complexity of RCR is $(l_M - 1)(O(n^2) + O(n^2)) + O(l_S n \log n)$. Therefore the time complexity of RCR is $O((l_M - 1)n^2)$.□

It is worth noting that the runtime complexity of IDM-kMDC is $O(n^4 \log n)$ (Senel and Younis, 2012), which is significantly higher than RCR. In comparison, in MiMSI the determination of SPs has a runtime complexity of $O(n^4)$ (Senel and Younis, 2011). Kmeans++, the clustering algorithm used by MiMSI to form clusters has a runtime complexity of $2^{\Omega\sqrt{n}}$) (Arthur and Vassilvitskii, 2006) where 'n' is the number of segments and SPs. Note that in MiMSI, due to the randomness in choosing cluster centers, if $l_S < l_M$ and all the cluster centers lie on SPs then merging needs to be carried out to meet the l_S bound. The tour formation can be

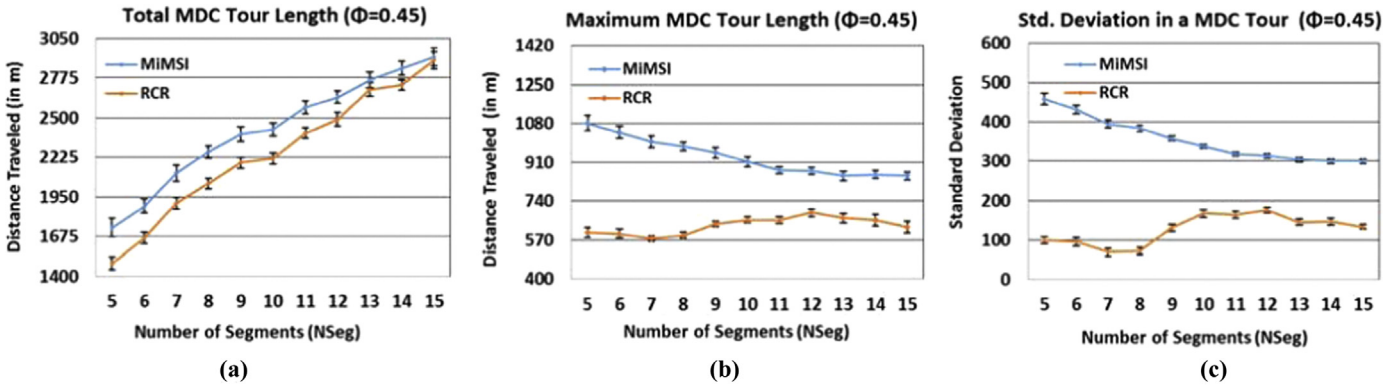

Fig. 12. Total distance traveled by MDCs for $\Phi=0.45$ and communication range $R=100$ m.

bounded by $O(n^2)$ and the optimization phase $O(l_s n \log n)$. Hence for MiMSI the runtime complexity can be bounded by $O(n^4)+2^{\Omega(\sqrt{n})}$, which is significantly more than RCR. In Section 5, we compare the performance of the RCR to both IDM-kMDC and MiMSI through simulation.

5. Performance evaluation

The effectiveness of RCR is validated through simulation. This section discusses the simulation setup, performance metrics and results.

5.1. Performance metrics and experiment setup

The experiments are conducted in a $1500 \times 1500\ m^2$ area wherein random topologies are generated for a varying number of segments (5–25) and communication range R is fixed at 100 m for all RNs. The CGAL computation library (CGAL) was used to carry out all geometric computations. We consider the following metrics to measure performance:

- *Total Tour length* : is the sum of tour lengths of all RNs employed as MDCs by RCR. A smaller total tour length implies lower overhead on the MDCs.
- *Maximum Tour length* : is the length of the longest tour amongst all MDC tours. A large value implies higher travel overhead on a MDC and increased data latency.
- *Standard Deviation* : measures the dispersal of tour lengths from the average tour length. A low value indicates that the tour lengths are very close to the average and the travel load is being shared equitably amongst the MDCs. A large value indicates unbalanced tours, which can result in poor data latency and cause losses due to buffer overflows in segments being covered by the longest tour. An overloaded MDC is also prone to running out of battery life quicker.

We study the performance while varying the following parameters:

- *Ratio of available RNs* (Φ) : Since RCR aims to solve the recovery problem when the surviving segments do not have enough RNs to form a stable inter-segment topology, we use Φ to control the RN count. The total number of available RNs is set to the number of Steiner points found in the SMT-MSPBEL multiplied by Φ, whose value is set at 0.45 and 0.55 respectively. Out of the available RNs, the ratio of l_M mobile to l_S stationary RNs is fixed at 1, i.e., $l_M/l_S=1$.
- *Number of Segments* (N_{seg}) : A higher number of disjoint segments in a WSN introduces more connectivity requirements between them and highlights the complexities of tour formation.

5.2. Simulation results

This section discusses the obtained results. Each configuration is averaged over 50 random topologies. We observed that with a 90% confidence interval, the results stayed within 5–10% of the sample mean.

In the first set of experiments we compare RCR with MiMSI (Abbas and Younis, 2013), which also tackles the connectivity restoration problem using a mix of MDCs and stationary RNs. In these experiments the communication range of the RNs is fixed at 100 m. The graphs in Figs. 12 and 13 show the variation in the total distance traveled by all MDCs, the maximum overhead on a MDC, and the standard deviation of the MDC tours as the number of segments are varied from 5 to 15. We also vary Φ, to highlight the effect increased resources have on connectivity restoration.

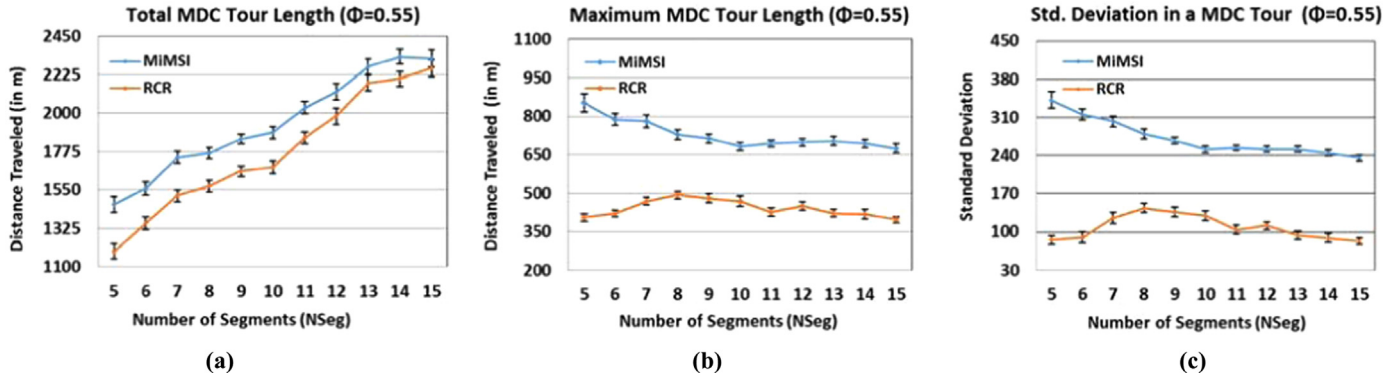

Fig. 13. Total distance traveled by MDCs for $\Phi=0.55$ and communication range $R=100$ m.

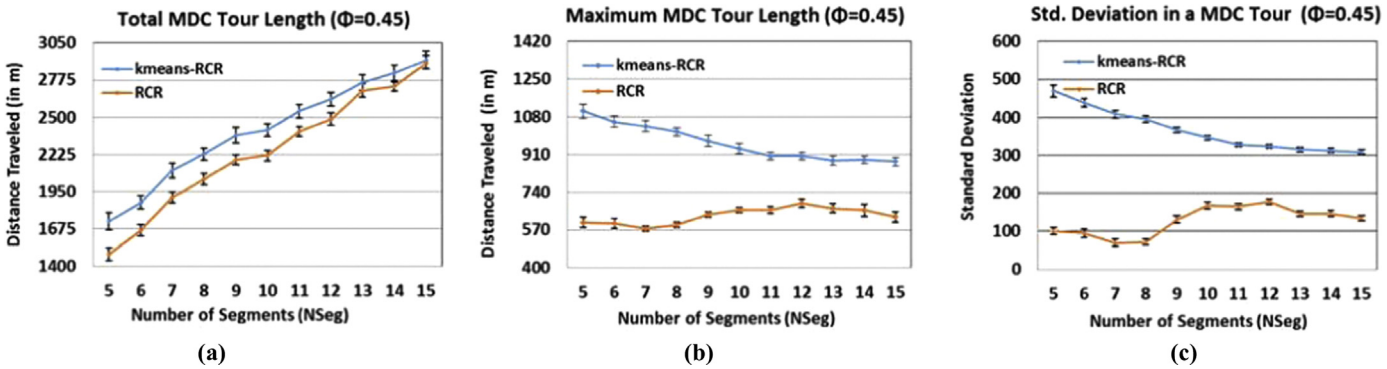

Fig. 14. Comparing the clustering method of RCR with MiMSI for Φ=0.45 and communication range R=100 m.

Figs. 12(a) and 13(a) show that RCR has a lower combined total tour length than MiMSI. This improvement is due to better segment grouping, which results in balanced tours and also by having a superior tour formation strategy which reduces the MDC travel overhead. We also see that as Φ increases the performance advantage for RCR grows larger, this is because larger Φ implies more MDCs and stationary RNs are available, thus more tours are formed and the advantages of RCR tour formation add up, amplifying the saving and resulting in a larger gap with respect to MiMSI. We see that the gap in total tour length decreases with the increase in the number of segments because the inter-segment proximity increases and the drawback of MiMSI's clustering algorithm diminishes. Recall that unlike RCR, the kmeans++ algorithm used by MiMSI does not factor inter-cluster distance. Thus, the increased segment density slightly improves the clustering performance in MiMSI.

Figs. 12(b) and 13(b) show that RCR handily outperforms MiMSI by having a much lower maximum tour load on a MDC, while Figs. 12(c) and 13(c) indicate that the standard deviation is also much lower for RCR generated MDC tours. A lower standard deviation in tour length is beneficial because it means that the tours are approximately equal in size. Having balanced tours means that no one MDC is overburdened into covering a larger area, which can make it run out of energy quicker than other MDCs. Obviously, losing an MDC due to energy exhaustion would mean re-calculating MDC tours again which is an additional overhead.

As Φ is varied from 0.45 to 0.55 we see a reduction in the total tour length and maximum tour length (Figs. 12 and 13). This is because an increase in Φ means availability of more resources, i.e., MDCs and stationary RNs to cover the same WSN. Increased RN availability means that the segment groups will be smaller since more MDCs are available and the extra stationary RNs can be utilized in the tour optimization phase to reduce the longest MDC tours.

To show that our segment grouping technique is superior to that of MiMSI, we run a set of experiments where we compare two versions of RCR, one utilizing kmeans++ clustering, like MiMSI, and the other applies our segment grouping technique. The tour optimization and formation procedure is kept the same in both cases. The results in Fig. 14(a) and (b) show a reduction in total tour length for the kmeans++ based RCR compared to MiMSI; such a reduction is attributed to our tour formation technique. However, from Figs. 14 and 15 it is clear that RCR's graph center based segment grouping is superior in all metrics like the total tour length, maximum overhead on a MDC and standard deviation amongst MDC tours. In addition to providing better results it is also much faster than kmeans++ which has super-polynomial time complexity, as we discussed in Section IV.

Finally, we compare the performance of RCR with recovery solutions that deal with the general case in which all available RNs can be utilized as MDCs. For this comparison we choose IDM-kMDC, the best performing approach that aims to minimize the overall MDC tour length. IDM-kMDC is a greedy approach, that forms an *mst* over the set of disjoint segments and assigns an MDC to each *mst* edge, and keeps combining tours that have the least merging cost until the number of tours meets the MDC availability constraint. Since IDM-kMDC considers all RNs to be mobile, to keep the comparison consistent we have set the number of stationary RNs to zero (l_S=0) and run the experiment for a fixed number of MDC (l_M=4) while varying the number of segments between 11 and 25. The number of MDCs is kept less than the number of segments in order to engage the tour formation algorithm of IDM-kMDC. The communication range R is fixed at 100 m. Fig. 16(a) shows that all approaches have approximately the same total tour length. Fig. 16(b) and (c) clearly highlight that RCR forms more balanced tours, while having approximately the same total

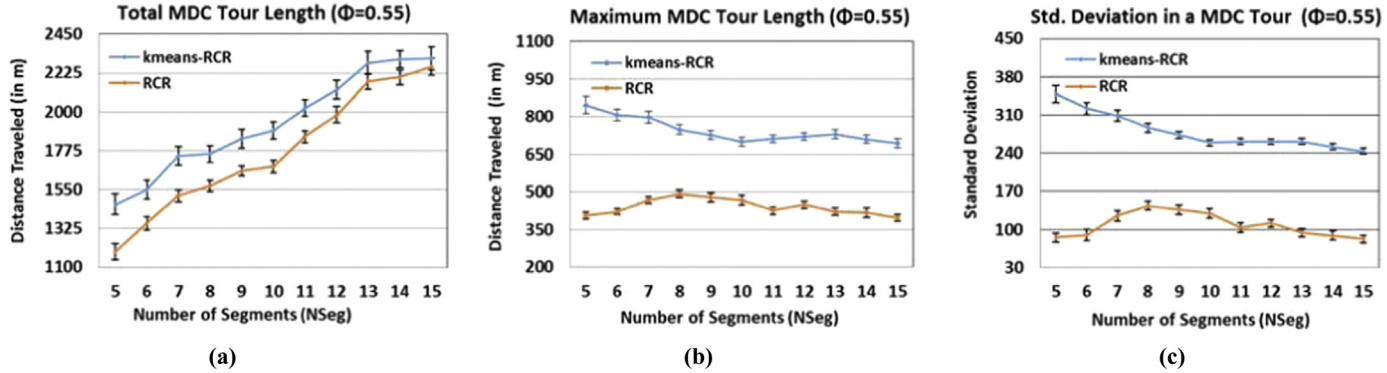

Fig. 15. Comparing the clustering method of RCR with MiMSI for Φ=0.55 and communication range R=100 m.

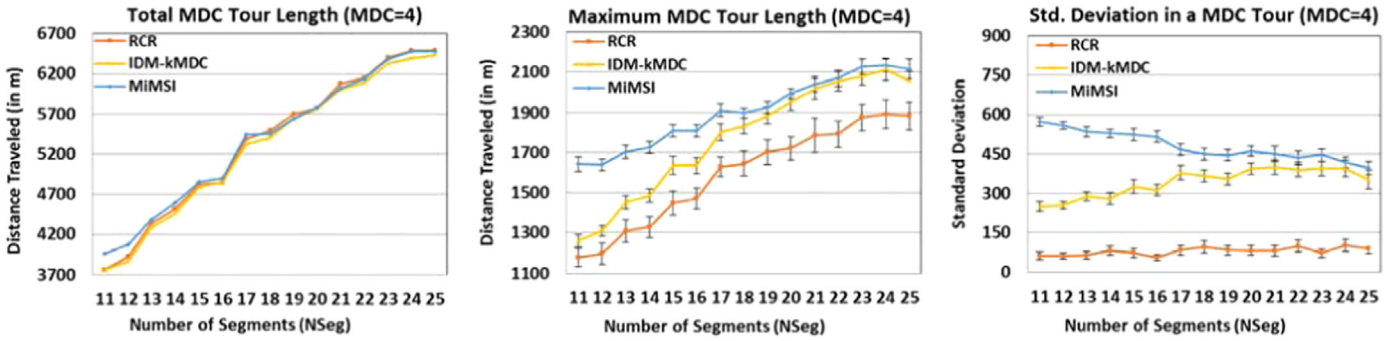

Fig. 16. Comparison of RCR with IDM-kMDC and MiMSI for fixed MDC count, namely, $lM=4$ and $lS=0$.

tour length as the competing approaches. RCR not only has the smallest maximum tour length amongst all three approaches but the standard deviation of the tours formed is also the lowest. This means that in RCR MDCs have a balanced work load and the data delivery latency will be much less in comparison with IDM-kMDC and MiMSI.

Overall, the simulation results in Figs. 12–16 confirm the performance advantage of RCR. RCR restores intermittent connectivity in a resource-constrained WSN and is shown to impose significantly less overhead on MDCs than competing approaches by utilizing a lightweight and effective segment grouping approach that produces balanced MDC tours. Also by exploiting the geometric properties of angle bisectors and polygons, RCR produces smaller tours over the same set of terminals as compared to other approaches. RCR segment grouping and tour formation reduce the total distance that MDCs have to travel for data exchange which translates into savings in energy and also improved the network lifetime. The performance advantage of RCR grows significantly as the percentage of RNs available grows since it means more MDCs are available and more tours are formed, thus increasing the gain in performance due to better segment grouping and tour formation.

6. Conclusion

An autonomous WSN should be able to recover from a large scale failure that damages multiple collocated nodes. In practical scenarios based on the level of damage the surviving segments in the network may not have enough resources to form a stable inter-segment topology. In this paper we have presented RCR, a novel distributed algorithm for restoring connectivity in a resource constrained WSN. RCR takes into account that not all available RNs can function as MDCs and some may have to be stationary due to low remaining energy or some external damage that impedes their movement. RCR achieves the recovery goal by initially relocating the fewest RNs from individual segments and then plans MDC tours and placement of stationary RNs to facilitate inter-segment connectivity. The simulation results have demonstrated that RCR scales well and outperforms competing approaches.

Acknowledgments

This work was supported by the National Science Foundation (NSF) award # CNS 1018171.

References

Abbas A, Younis Mohamed. Establishing connectivity among disjoint terminals using a mix of stationary and mobile relays. Comput. Commun. 2013;36 (13):1411–21.
Akkaya K, Senel F, Thimmapuram A, Uludag S. Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. IEEE Trans. Comput. 2010;59(2):258–71.
Almasaeid, H., Kamal, A.E., 2007. Data delivery in fragmented wireless sensor netowrks using mobile agents. In: Proceedings of the MSWiM 07, Chania, Greece.
Arthur, D., Vassilvitskii, S., 2007. k-means++: the advantages of careful seeding. In: Proceedings of the 18th annual ACM-SIAM symposium on discrete algorithms (SODA'07), Philadelphia, PA.
Alsalih, W., Akl, S.G., Hassanein, H.S., 2007. Placement of multiple mobile base stations in wireless sensor networks. In: Proceedings of the ISSPIT'07, Cairo, Egypt.
Alsalih W, Hassanein H, Akl S. Placement of multiple mobile data collectors in wireless sensor networks. J. Ad Hoc Netw. 2010;8:378–90.
Applegate DL, Bixby RE, Chvátal V, Cook WJ. The Traveling Salesman Problem: A Computational Study. New Jersey, USA: Princeton University Press; 2006.
Arthur, D., Vassilvitskii, S., 2006. How slow is the k-means method? In: Proceedings of the SCG'06, Sedona, AZ.
CGAL, Computational Geometry Algorithms Library. ⟨http://www.cgal.org⟩.
Chen, B., et al., 2001. Span an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In: Proceedings of the ACM MobiCom'01, Rome, Italy.
Cheng X, Du D-z, Wang L, Xu B. Relay sensor placement in wireless sensor networks. Wirel. Netw. 2008;14(3):347–55.
Das, S., et al., 2007. Localized movement control for fault tolerance of mobile robot networks. In: Proceedings of the 1st IFIP International Conference on Wireless Sensor and Actor Networks (WSAN 2007), Albacete, Spain.
Han, X., Cao, X., Lloyd, E.L., Shen, C.-C., 2007. Fault-tolerant relay nodes placement in heterogeneous wireless sensor networks. In Proceedings of the INFOCOM'07, Anchorage AK.
Jain S, Shah RC, Brunette W, Borriello G, Roy S. Exploiting mobility for energy efficient data collection in wireless sensor networks. J. Mob. Netw. Appl. 2006;11:327–39.
Joshi, Y.K., Younis, M., 2012. Autonomous recovery from multi-node failure in wireless sensor network. In: Proceedings of the Globecom'12, Anaheim, CA.
Joshi, Y.K., Younis, M., 2013. Distributed approach for reconnecting disjoint segments. In: Proceedings of the Globecom'13, Atlanta, GA.
Joshi, Y.K., Younis, M., 2014. Straight skeleton based reconnection in a wireless sensor network. In: Proceedings of the Globecom'14, Austin, TX.
Joshi, Y.K, Younis, M., 2014. Mobility-based internetworking of disjoint segments. In: Proceedings of the 27th biennial symposium on communications (QBSC), Kingston, Ontario, Canada.
Joshi YK, Younis M. Exploiting skeletonization to restore connectivity in a wireless sensor network. Comput. Commun. 2015.
Lee S, Younis M. Recovery from multiple simultaneous failures in wireless sensor networks using minimum steiner tree. J. Parallel Distrib. Comput. 2010;70:525–36.
Li, N., Hou, J.C., 2004. Flss: a fault-tolerant topology control algorithm for wireless networks. In: Proceedings of the MobiCom'04, Philadelphia, PA.
Lin G, Xue G. Steiner tree problem with minimum number of Steiner points and bounded edge-length. Inf. Process. Lett. 1999;69(2):53–7.
Lloyd EL, Xue G. Relay node placement in wireless sensor networks. IEEE Trans. Comp. 2007;56(1):134–8.
Shah, R., Roy, S., Jain, S., Brunette W., 2003. Data MULEs: modeling a three tier architecture for sparse sensor networks. In: Proceedings of thr IEEE Int' workshop on sensor network protocols and applications (SNPA'03).
Senel F, Younis M. Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation. Elsevier Comput. Commun. 2011;34(16):1932–41.

Senel, F., Younis, M., 2012. Optimized relay node placement for establishing connectivity in sensor networks. In: Proceedings of the Globecom'12 Anaheim, CA.
Senel, F., Akkaya, K., Younis, M., 2007. An efficient mechanism for establishing connectivity in wireless sensor and actor networks. In: Proceedings of the Globecom'07, Washington, DC.
Shen, C.-C., Koc, O., Jaikaeo, C., Huang, Z., 2005. Trajectory control of mobile access points in MANET. In: Proceedings of the Globecom'05, St. Louis, MO.
Senel, F., Younis, M, 2012. Optimized interconnection of disjoint wireless sensor network segments using K mobile data collectors. In: Proceedings of the Inernational Conference on Commerce (ICC'12), Ottawa, Canada.
Vemulapalli, S., Akkaya K., 2010. Mobility-based self route recovery from multiple node failures in mobile sensor networks. In: Proceedings of the WLN'10, Denver, CO.
Wang, G., Cao, G., La Porta, T., Zhang, W., 2005. Sensor relocation in mobile sensor networks. In: Proceedings of the INFOCOM'05, Miami, FL.
Wang, X., Xing, G., Zhang, Lu, Y. C., Pless, R., Gill, C., 2003. Integrated coverage and connectivity configuration in wireless sensor networks. In: Proceedings of the ACM Sensys'03, Los Angeles, CA.
Wuchty S, Stadler Peter F. Centers of complex networks. J. Theor. Biol. 2003;223 (1):45–53.
Younis M, Senturk I, Akkaya K, Lee S, Senel F. Topology management techniques for tolerating node failures in wireless sensor networks: a survey. Comput. Netw. 2014;58:254–83.