Dynamic Clustering and Management of Mobile Wireless Sensor Networks

Abdelrahman Abuarqoub[a,*], Mohammad Hammoudeh[b], Bamidele Adebisi[b], Sohail Jabbar[c], Ahcène Bounceur[d], Hashem Al-Bashar[a]

[a]*Faculty of Information Technology Middle East University, Airport Road Amman 11831, Jordan*

[b]*Manchester Metropolitan University, Chester Street, Manchester M1 5GD, UK*

[c]*School of Computer Science and Engineering, Kyungpook National University, Korea*

[d]*Computer Science Department, University of Brest, Brest 29238, France*

Elsevier use only: Received date here; revised date here; accepted date here

Abstract

In Wireless Sensor Networks (WSNs), routing data towards the sink leads to unbalanced energy consumption among intermediate nodes resulting in high data loss rate. The use of multiple Mobile Data Collectors (MDCs) has been proposed in the literature to mitigate such problems. MDCs help to achieve uniform energy-consumption across the network, fill coverage gaps, and reduce end-to-end communication delays, amongst others. However, mechanisms to support MDCs such as location advertisement and route maintenance introduce significant overhead in terms of energy consumption and packet delays. In this paper, we propose a self-organizing and adaptive Dynamic Clustering (DCMDC) solution to maintain MDC-relay networks. This solution is based on dividing the network into well-delimited clusters called Service Zones (SZs). Localizing mobility management traffic to a SZ reduces signaling overhead, route setup delay and bandwidth utilization. Network clustering also helps to achieve scalability and load balancing. Smaller network clusters make buffer overflows and energy depletion less of a problem. These performance gains are expected to support achieving higher information completeness and availability as well as maximizing the network lifetime. Moreover, maintaining continuous connectivity between the MDC and sensor nodes increases information availability and validity. Performance experiments show that DCMDC outperforms its rival in the literature. Besides the improved quality of information, the proposed approach improves the packet delivery ratio by up to 10%, end-to-end delay by up to 15%, energy consumption by up to 53%, energy balancing by up to 51%, and prolongs the network lifetime by up to 53%.

Keywords: Wireless sensor network routing; Clustering protocol; Mobile data collectors; Dynamic network clustering; Mobility management; Self-organisation.

1. Introduction

In the last two decades, Wireless Sensor Networks (WSNs) have significantly changed the way we interact with our environment. WSNs appear in several scientific, commercial, health, surveillance, and military applications. Unattended sensor nodes offer maintenance-free operation and can detect, characterize and disseminate situational awareness continuously. After years of research on the opportunities provided by WSNs and their potential value, many of those looking to operate in the WSN market are starting to consider the potential practical problems, including data management. Once a WSN is up and running at full scale, it will generate large quantities of data that need to be processed and analyzed in real time.

The success of WSN applications is dependent on knowing that information is available, the type of information, its quality, its scope of applicability, limits of use, duration of applicability, likely return, cost to obtain and a host of other essential details. To aid in data collection, the use of mobile nodes has been widely suggested in the literature. Node movement can be controlled and optimized to improve data collection and analysis. For instance, mobile nodes can be used to bridge disconnected parts of the network. Furthermore, node mobility can optimize the energy consumption and lifetime of a WSN. For example, moving the sink to data sources or moving the sensor nodes towards the sink is one way to avoid the communication bottlenecks. However, the deployment of mobile nodes instigates frequent topological changes that need to be resolved before data collection can be resumed.

In this paper, we propose a holistic self-organizing mechanism that is adapted from [1]. The proposed mechanism is based on clustering the network in cooperating zones. This is achieved by abstracting the network to a three-tier pyramid model as shown in Fig. 1. Each tier contains a different class of sensor nodes. The bottom tier hosts static nodes, which form the majority of the network population. Sensor nodes at this tier perform sensing and communication tasks. The middle tier hosts a small number of resource-rich mobile nodes, called Mobile Data Collectors (for short, MDCs). MDCs have long-range radio and are considered power rich devices. The top tier of the hierarchy hosts the fixed data sink(s). At this tier, information received from different sources is processed and presented to end users.

It is desirable to design a distributed and self-organizing strategy featuring adaptability to network topology changes to reduce the cost of topology updates. Higher energy efficiency can be achieved by reducing the frequency of connectivity disconnections. Fewer and shorter disconnections results in reduced signaling overhead and lower packet loss.

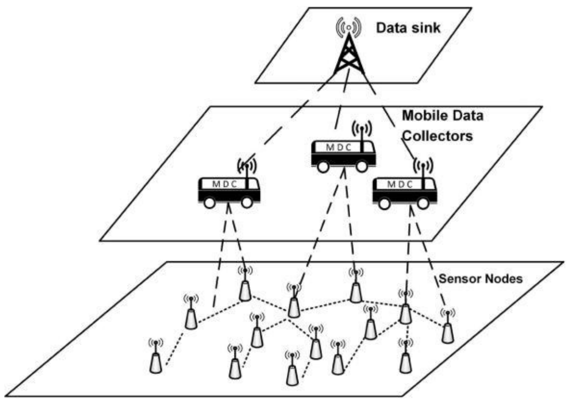

Fig. 1. A Three-tier network structure.

The mobility management protocol proposed in this paper assumes that nodes have knowledge of their physical location. Nodes can determine their location in several ways. The Global Positioning System (GPS) is the simplest localization method. GPS modules are known for their high energy consumption. They also increase the total cost of each sensor nodes. Currently, the CRIUS MultiWii MWC I2C-GPS module costs less than \$5. GPS-based solutions can become very expensive in large-scale WSN deployments. Therefore, several localization algorithms that do not relay on GPS modules have been introduced in the literature, e.g., [2, 3]. Localization algorithms use various information available from the network in order to calculate the correct position of each sensor node. There are several localization techniques for indoor and outdoor environments described in the literature [4-6]. The use of localization algorithms introduces additional challenges that need to be addressed, including: (1) Localization latency: the localization algorithm should take minimal time to cope with mobility speed; (2) Increased control messaging: managing nodes location information requires communications and transmission of control packets. In mobile WSN where nodes change their location frequently, the localization control overhead will increase significantly causing network congestion and leading to higher energy consumption. In the proposed system, we assume that all MDCs and a small percentage of nodes are equipped with GPS modules. Other nodes in the network can estimate their location by using a centroid formula, where anchor nodes transmit their location to the blind nodes. This method keeps the system cost low, while maintaining low localization overhead [2, 3, 5].

The remainder of this paper is organized as follows: Section 2 examines background and related work in the area of MDCs in WSNs. Section 3 briefly introduce each phase of the proposed dynamic clustering technique, DCMDC. Section 4 presents the details DCMDC processes and phases. Section 5 discusses how DCMDC handles orphaned nodes. Section 6 presents the evaluation results of DCMDC and compares them against two of its best rivals in the literature. Section 7 concludes the paper.

2. Related Work

Network clustering has been widely investigated by the WSN research community in the past two decades. In such clustering schemes, sensor nodes are divided into multiple logical groups according to some rules. These rules may be relate to a node's deployment, capabilities or other network dynamics [7]. The literature is very rich with effective clustering approaches designed for WSNs. The work in [8] focused on preserving complete coverage of the monitored area over long periods. The authors in [9] proposed a cluster based algorithm for tracking a mobile target to achieve high tracking accuracy and energy efficiency. Other propositions exist in the literature devoted to study mobility estimation and mobility supporting protocols in WSNs; a recent schedule-based MAC protocol for static and mobile nodes is investigated in [10].

Several hierarchical architectures have been considered for various applications of mobile WSNs [7, 11-15]. In some approaches, cluster heads are used as MDCs besides their sensing duties. MDCs are used to carry information from the sensing field and deliver it to a fixed sink. In these approaches, sensor nodes send data over short-range communication, from a sensor to the MDC, which requires less transmission power due to the reduced bridging distance between data sources and the sink. MDCs also avoid the effect of bottlenecks, especially in areas around the sink, such as packet loss, increased end-to-end delay and energy depletion. The existence of multiple data collectors reduces the breakdown of interconnections; meaning that if one data collector fails, data can be transmitted through another data collector.

Although using MDCs is desirable due to their simplicity and efficiency, they introduce major challenges. Managing MDC location information requires communication and transmission of control packets. When the location changes frequently, the control packet overhead will increase, which leads to higher energy consumption. This may possibly dissipate the energy gains achieved by the MDCs. Moreover, the movement of MDCs may introduce significant data delivery delay caused by link establishment time, velocity control, etc. Finally, the MDC travel trajectory calculation is a complex problem.

There are several approaches devoted to the study of hierarchical mobile WSNs. Energy efficient routing protocols for multiple MDCs are investigated in [16, 17]. The placement and relocation of multiple MDCs is investigated in [18]. Data collection approach to support mobility with multiple MDCs is presented in [19]. Secure cluster head election, where the cluster head is not a malicious node, is presented in [20]. However, there is only a handful set of papers directly addressing the problem of relay nodes mobility management.

In [21], the author proposed an Energy-efficient Cluster-based Data Gathering Algorithm (ECDGA) for mobile WSNs. The network model of ECDGA consists of heterogeneous sensor nodes. Static nodes are deployed in a grid to manage dynamic changes in the topology and relay sensed data from nearby cluster heads to a slow-moving sink. The cluster head selection is based on the residual energy and location of the mobile nodes. The authors show that ECDGA effectively prolongs the network lifetime. Nonetheless, ECDGA algorithm does not consider mobility parameters such as mobility speed and direction when allocating mobile nodes to clusters.

In [22], the authors proposed a self-organization method for mobile devices in cluster-based ad-hoc networks. This method is implemented through a multi-role agent approach. Each agent could be a leader, gateway or member; the roles assignment is based on the remaining energy in the node and its neighborhood. When the network is deployed, a role assignment process takes place. When the remaining energy in the leader agent reaches a certain threshold, it reduces its transmission range to and when it reaches a lower threshold, the leader election procedure is executed. However, the strategy establishes too many leaders in the network, which causes bandwidth wastage, and a large number of collisions. Furthermore, the sensor node weight function considers only the residual battery and the number of neighbors. It does not consider the node position, mobility speed and direction.

In [23], a hybrid multipath routing algorithm with an efficient clustering technique is proposed. The algorithm uses an energy-aware selection mechanism to choose the fusion nodes to route the data to a data sink. A node is chosen to play the role of a fusion node if it has high level of energy, high transmission range and lower mobility. The network is divided into multiple square zones, each square is considered a cluster that is managed by a selected fusion node. The square zones allow the union of zones without holes, and simplify the design of clustering algorithm. However, the mobility metric is calculated as the measure of relative motion of nodes. The mobility measure is normalized by the number of nodes and the continuous functions of time that represents the quantitative measures of relative motion between nodes.

Recently, Battery-Level Aware Clustering (BLAC) was presented in [24]. BLAC considers the battery-level combined with another metric to elect the cluster head. It comes in four variants: BLAC-bg combines battery level and node degree, BLAC-bs uses the battery level and node density, BLAC-rg and BLAC-rs. The last two variants run in two steps. They first apply graph reduction followed by network clustering. Each of these variants presents specific features that make them more suitable than others under different conditions. If nodes are mobile, BLAC-bs is the best choice, as it offers a better stability against mobility.

More recently, the anthers of [25] proposed Efficient Routing Protocol for Multiple Mobile Sink Based Data Gathering (ERMMSDG). This protocol uses a biased random walk method to estimate the next position of the MDC. To determine the optimal data transmission path, a rendezvous point selection with splitting tree technique is used. Whenever the sink passes through the rendezvous point, it receives the collected data. Alternatively, a relay node from its neighbors relays packets from rendezvous point to the sink. The protocol reduces the signal overhead and improves the triangular routing problem. However, the relay node selection mechanism does not consider the mobility angle and speed of the mobile collector; it only considers the distance to the mobile collector. This could lead to routing the data through a longer path, introducing considerable delays on data delivery and bandwidth wastage. Furthermore, the relay node selection mechanism does not consider the residual energy of the relay node, which may lead to network segmentation, and unbalanced energy in the network.

Similar to ERMMSDG, MDC/PEQ [26] is an approach that uses mobile data collectors to achieve low-latency and reliable mobile data gathering in delay-sensitive applications. In MDC/PEQ, MDCs broadcast configuration beacons periodically. Initially, when a sensor node receives a beacon, it joins the MDC's cluster and updates its routing information accordingly. For connection reconfigurations (handoffs), sensor nodes use the signal strength of the beacon as well as the number of hops to reach the MDC. Each node holds two communication paths, one is a multi-hop direct link to the sink, and the other is to an MDC. When a node has data to be transmitted, it uses the shorter path, i.e., the path with the smaller number of hops. If the both paths have the same number of hops or the node is not associated with an MDC, the node relays its data through the direct path to the sink. If an intermediate node has a route to an MDC, it forwards the data to that MDC. This approach achieves good timeliness as nodes do not wait for an MDC to move nearby. However, MDC/PEQ produces high MDC advertisement overhead that need to be received, processed and forwarded. Furthermore, it uses the path with the smaller number of hops, which is not necessarily the most reliable or energy efficient. Finally, sensor nodes that do not belong to any cluster use the direct route to the sink to transfer their data. This leads to conveying data through several hops, thereby contributing to an increase of packet collisions and losses.

There is a wide body of literature on clustering algorithms designed for WSNs. We refer interested readers to [27, 28], recent surveys that provide a comprehensive review to data collection approaches designed for static and mobile WSNs. Most of the reviewed approaches are proved effective and efficient. However, there are few attempts to address the problem of relay node mobility management [29-31]. These attempt to deal with mobility as the need arises and do not deal with the fundamental challenges and variations introduced by mobility on the WSNs. We believe that there is a need for a holistic self-organizing strategy that organizes MDCs in such a way that signaling overhead is reduced, while keeping energy consumption and resource usage to the minimum. Additionally, such a strategy should take into consideration MDC mobility parameters such as mobility direction and speed.

3. Dynamic Clustering for Mobility Management

In this section, we introduce a self-organizing and adaptive Dynamic Clustering (DCMDC) solution to effectively manage topological updates and maintain communication routes in mobile WSNs. Aiming at reducing mobility-triggered signaling overhead, we design and implement a dynamic self-organizing protocol that partitions the network to a set of well-delimited logical network clusters called Service Zones (SZs). The dynamic clustering of the network into SZs is based on the convex hulls algorithmic problem. Organizing the network in SZs offers several advantages. Firstly, it reduces signaling overhead, consequently bandwidth utilization, by localizing mobility management traffic. It is well established in the literature that less congestion reduces queuing delays. Secondly, network clustering is a well-tested solution to achieve scalability and load balancing. Grouping nodes into smaller logical sets makes buffer overflows and energy depletion less of a problem. Finally, maintaining continuous connectivity with the MDC when it is in the communication range of sensor nodes increases the system availability.

In dynamic network clustering, the frequent boundary updates presents a significant challenge. Collection Zones (CZs) are introduced to tackle this challenge. CZs are designed such that their maintenance is quick and efficient. CZ maintenance cost is reduced by performing efficient neighbor discovery and localized computation. The CZ of an MDC is defined by the set of nodes directly connected to that MDC. The movement of an MDC within its defined CZ does not require connectivity or neighborhood update. This enables nodes within a CZ to view their MDC as a virtually static node for a certain period. Depending on its speed, sensor nodes can easily predict the connectivity period with their present MDC.

The result of the network clustering process is based on the number of MDCs, their positions and their movements. For each MDC, a convex group of nodes that will form the MDC's SZ is defined. The construction of convex groups does not exhibit high computational complexity [32]. While an MDC is moving inside its SZ, it performs several operations to keep the network topology up to date. These operations include, updating its SZ members list, connecting new nodes or disconnecting existing nodes from its SZ. These operations ensure that sensor nodes can be allocated to the best MDC that can forward its data to the sink more effectively. Simultaneously, sensor nodes should be allocated to MDCs to load balance their workload.

Fig. 2 shows the conceptual relationship between the SZ and the CZ. A SZ is a designated geographical zone around an MDC, containing a set of nearby sensor nodes. The SZ forms a convex group of nodes constructed by the MDC. The MDC is responsible for all communication in that SZ. MDCs exchange control messages with each other and only with sensor nodes that belong to their SZ. In this way, every sensor node receives control messages only from the MDC that it belongs to. As a result, flooding problems from MDCs to sensor nodes in the bottom layer are avoided.

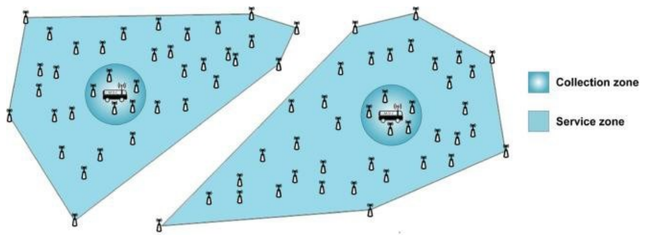

Fig. 2 - The logical structures defined around an MDC

A CZ of an MDC is a circular area around an MDC with radius equal to half of the radio range of sensor nodes. In Cartesian coordinates, the center of the CZ is the physical position of the corresponding MDC before it moves. An MDC communicates directly with nodes that are inside its CZ. An MDC can move inside its CZ and stay directly connected to the same set of nodes. This design exploits the fact that the active radio coverage of MDCs is wider than their CZs. Therefore, sensor nodes can consider the MDC static until it moves out of its CZ. Hence, it does not need to issue any neighbor discovery or update messages during this period. SZs are assumed larger than the CZs. The SZ and the CZ change dynamically depending on the MDCs speed and direction information. The position, speed and direction of the nodes can be obtained by a GPS device providing latitude, longitude, altitude, speed and travel track.

Initially, directly after network deployment, DCMDC runs through three phases: neighborhood discovery, CZ creation and network formation. Fig. 3 illustrates the various phases of the network setup process. During the neighborhood discovery phase, MDCs create binding tables for storing nodes' information. MDCs proactively advertise their presence to neighboring sensor nodes who choose the optimal MDC to join. During the CZ creation phase, each MDC constructs its CZ boundaries and determines which sensor nodes are located inside it. These sensor nodes will have direct communication with the MDC. During the network formation phase, reconciliation of the overlapped CZs occur and SZs are formed.

The next section goes through the detailed DCMDC processes of setting up the network to prepare it for operation. The details of each setup phase are presented and discussed.

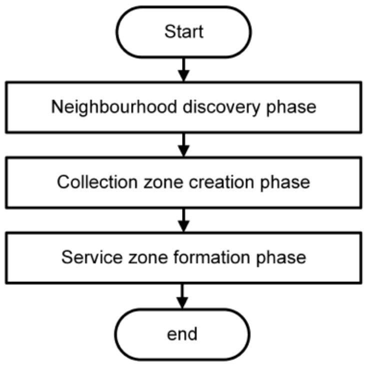
Start
Neighbourhood discovery phase
Collection zone creation phase
Service zone formation phase
end

Fig. 3 - Network setup procedure.

4. DCMDC Protocol Details

4.1. Neighborhood Discovery Phase

In the literature, neighborhood discovery can be classified into the following main categories: proactive, e.g., [33], reactive, e.g., [34] and hybrid, e.g., [35, 36]. In the proactive approach, the data collector periodically broadcasts an advertisement message. When the message is received by a sensor node, that node creates a route to the data collector and relays the advertisement message to its neighbors. This results in many duplicated messages consuming valuable bandwidth and energy. In contrast, in the reactive approach, discovery messages to initialize or update connections are initiated by sensor nodes. The sensor node broadcasts a connection request message in the network. When a data collector receives the message, it unicasts a reply message containing its details (for example, its location and available resources or services). This approach saves bandwidth and energy as it sends requests only when information is needed. However, the main drawback of this approach is the high latency in data collector discovery and bottlenecks around the data collectors. The hybrid approach uses a combination of the two above approaches by considering the disadvantages of both of them.

Sensor nodes in DCMDC use a hybrid MDC discovery approach to adapt to various network conditions. Before joining a SZ, a sensor node uses a proactive MDC discovery approach to identify the optimal MDC to associate themselves with. After the construction of SZs, orphaned nodes use a reactive MDC discovery approach to participate in the network. The details of how DCMDC deals with orphaned nodes are given in Subsection 5. Every MDC maintains a binding table to store information about its CZ membership. Table 1 shows the content of the binding table with example values. The binding table is used to store information about sensor nodes that are connected to the MDC.

Table 1 Binding table entry example

Node ID	X position	Y position	Track	Battery level	Last update time
1					

(1)

As illustrated in Fig. 4, every MDC proactively discovers its neighbors. It broadcasts advertisement messages containing its location information. When a sensor node receives a packet, it sends it to all neighbors, which results in significant redundancy, collisions and contention. To reduce the impact of such consequences, MDCs broadcast advertisement messages to all nodes that are closer than the maximum distance, d, over which an advertisement message can be transmitted. On receiving the message, each sensor node makes a decision about the optimal MDC with which to associate itself. The decision is based on the Optimal Sink Selection algorithm (OMSS) published in [37]. OMSS is based on a parameter called the Connection Expiration Time (CET). In the following, a brief explanation of the OMSS decision algorithm is provided.

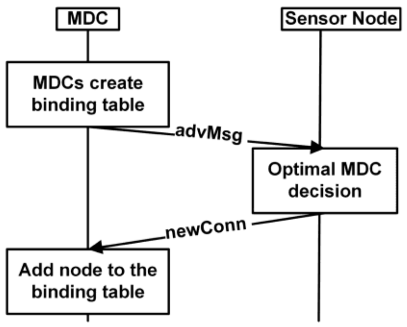

Fig. 4 - Neighborhood discovery messages diagram.

Let be a sensor node that received an message from an MDC. The MDC moves in the direction in two-dimensional space with respect to the positive X-axis. Let be the location of a sensor node and be the location of the MDC. Suppose that the MDC travels at the speed of . The velocity of the MDC on the and axis can be calculated using the equations (2) and (3), respectively.

(2)
(3)

To calculate the CET, we use equation (1) that factors the location of the MDC, its movement speed and direction from a sensor node, link reliability and available resources. In equation (1), is a constant of proportionality for the workload adjustment. is the maximum distance that the MDC forward advertisement message over.

The link reliability is measured in terms of the weighted average of the probability of successful packet reception by an MDC from node . Because these communication links are bidirectional, we consider the weighted average of probabilities of both transmission directions. is defined as:

(4)

Algorithm 1 presents the steps followed by a sensor node to determine the optimal MDC. maxConnection is defined as the remaining connectivity time to the current MDC. When a sensor node is not connected to an MDC, it waits for a short period to allow for advertisement from all MDCs in its vicinity to arrive. Then, it joins the MDC that offers the highest CET value. If a sensor node, which is currently associated to an MDC, receives an advertisement message with a better CET value, then it leaves the current MDC and joins the new one. If the node is within the vicinity of multiple MDCs with similar CET value, then the node joins the MDC with the lowest workload level.

Algorithm 1: OMSS Algorithm

```
Input: MDC details, sensor node location
Begin
MDC_ID = MDC_1; maxConnection = 0
for every MDC MDCi do
  if CETMDCi > maxConnection then
     maxConnection = CETMDCi;
     MDC_ID = MDCi ;
  else
     if CETMDCi = maxConnection then
        if MembersNoMDCi < MDC_ID.MembersNo then
              MDC_ID = MDCi ;
        endif
     endif
  endif
endfor
return MDC_ID;
End
Output: ID of the selected optimal MDC
```

After making the decision, each sensor node replies with a message to the chosen MDC. Finally, MDCs receive replies from different nodes and add them to its binding table.

4.2. Collection Zones Creation Phase

One solution to minimize neighborhood updates is to predict when a node is expected to leave the SZ. The basic and simple way for neighborhood maintenance is by using periodic discovery messages. However, the most significant drawback for this method is choosing the rate at which the messages are sent. A high beacon rate results in increased bandwidth usage and communication cost. In contrast, a low beacon rate may possibly miss important topology changes or events where critical reconfigurations take place.

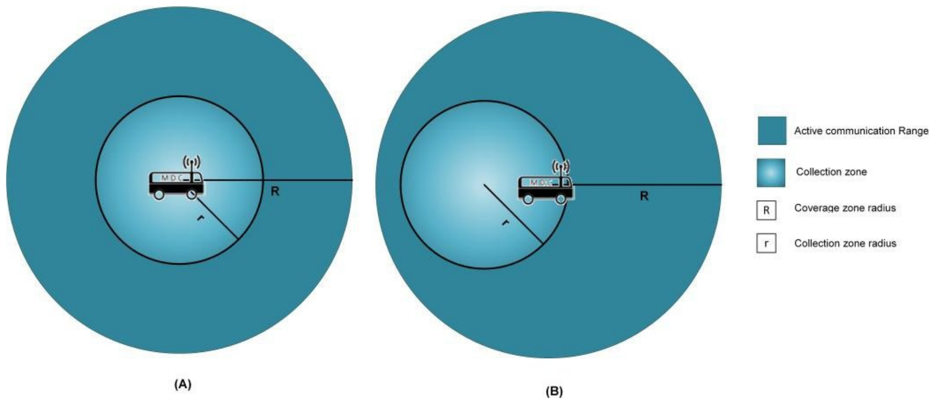

Fig. 5 - MDC coverage zone and collection zone.

In dynamic clustering of the network to convex hulls, updating convex envelope using messages consumes high bandwidth and energy. To determine when a convex hull update is necessary, we propose and define a CZ. CZs allow nodes to determine when to issue a neighbor discovery message and reconfigure their local connections.

Let be the set of sensor nodes within the MDC active communication range. Then, the CZ is defined as:

(5)

where is the distance between the sensor node and the MDC and is the active communication range of the MDC.

An MDC can move inside its CZ and stay directly connected to the same set of nodes. As long as the MDC is inside its CZ, it does not need to issue any neighbor update messages. In Fig. 5, the MDC defines the CZ as a smaller inner circle of radius in its active communication range. Initially, when an MDC creates its CZ, it will be located in the centre of the created CZ, Fig. 5-A. The MDC checks its binding table and determines which sensor nodes belong to its CZ (i.e.,) using Equation 6.

In Fig. 5-B, although the MDC has moved, no update is required as long as the MDC is inside its CZ perimeter. When an MDC leaves its CZ, the collection zone will be updated and discovery messages will be exchanged to reconfigure the network changes. The update process includes adding some nodes located inside its active communication range to the updated CZ, and removing nodes that belong to the original CZ. Some nodes that are already in the original CZ remain inside the updated one, i.e., the intersection area between the two collection zones in (see Fig. 5-B). The MDC does not exchange configuration messages with these nodes.

4.3. SZ Formation Phase

The SZs formation phase commences when MDCs have their CZs created. Throughout this paper, the terms convex hull and SZ are interchangeable. In mathematics, the convex hull of a set of points in the Euclidean plane is defined as the intersection of all convex sets containing or as the set of all convex combinations of points in . A set is said to be convex if for every pair of points within the set, every point on the line segment that joins the pair of points is also within the set.

Let be a set of sensor nodes, where is the minimum number of nodes to create a valid convex hull. Each node is represented as a pair . Let the function return a set of nodes that encloses the nodes in set , i.e., border nodes. is the smallest convex region that contains and is called the convex hull of the nodes set . If is the number of nodes in , then . The set stores the list of vertices of the convex hull in counter clockwise order.

MDCs use local information stored in their binding tables to construct their SZs. The vertices of the SZs will be the farthest connected nodes from the MDC. However, to maintain load balancing among various SZs, the SZs formation phase is composed of two steps: CZ reconciliation and SZ construction. The former step is only performed by MDCs that have overlapping in their CZs. The latter step is performed by all MDCs in the network.

(6)

4.3.1. Collection Zone Reconciliation

At the end of the CZ creation phase, a situation may arise where two or more MDCs have overlapping CZs. This situation can also occur after the SZ construction step if an MDC updates its SZ after moving to the vicinity of another MDC. These situations can result in creating small SZs that contain MDCs close to the perimeter of their corresponding SZ.

Consider the scenario in Fig. 6, where there are three MDCs that are physically close to each other and have overlapped CZs. In this case, each MDC constructs relatively small SZs. The MDCs will be located close to the perimeter of their SZs. Such situation is far from ideal for the following reasons: (1) It is possible that the MDC will very soon move outside its SZ. This results in a major SZs re-configuration at minor intervals, during which information delivery is interrupted. (2) Spatial events become more difficult to capture in a smaller SZ without high-level coordination. (3) Mobility management, because the update procedure runs within each SZ independently. To overcome the problem of over-clustering the network, we propose a CZ reconciliation algorithm, designed for choosing the appropriate MDC to serve the sensor nodes connected to the other MDCs.

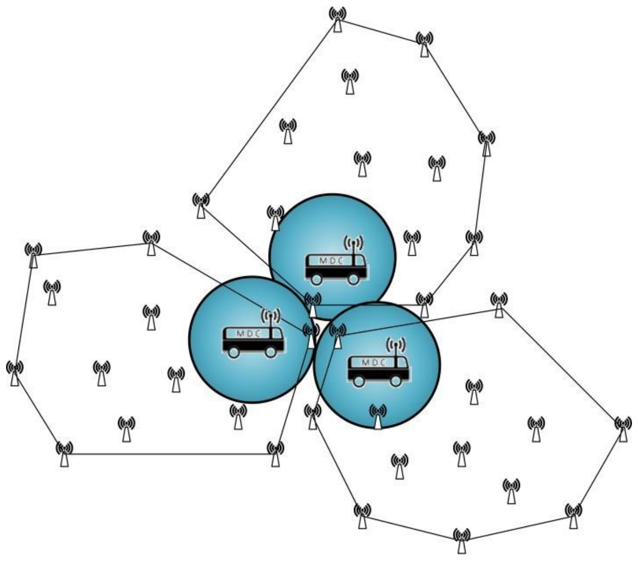

Fig. 6 - MDC merging situation.

Fig. 7 shows the steps MDCs follow to discover an overlap. MDCs check for CZs overlapping when it directly receives an advertisement message from another MDC. If the distance separating two MDCs is less than the length of their CZ diameter, then an overlap is detected. Upon CZ overlap detection, the discovering MDC sends an to the advertising MDC. Then, both MDCs execute the CZ reconciliation algorithm described below.

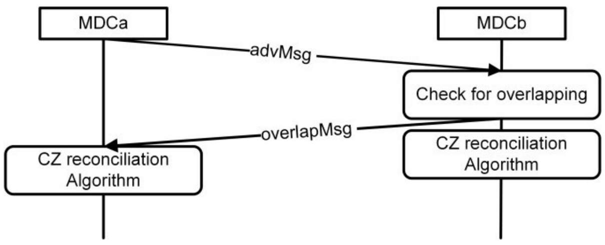

Fig. 7 - MDCs overlapping messages diagram.

Fig. 8 illustrates the CZs reconciliation algorithm. Candidate MDCs start by finding the MDC that has more members in its binding table. This MDC, called primary MDC, is chosen to form the new SZ. The members of other SZs will be transferred to the new MDC. The MDC with the greater number of members is retained to avoid the higher cost to transfer them to a different SZ. When two candidate MDCs have equal number of members, the MDC with the higher latitude is retained.

Fig. 9 is a step-by-step illustration of the messages exchanged during this process. The primary MDC sends a message to other involved MDCs. The receiving MDCs send messages to their members. Each member sends message to join the primary MDC. Upon receiving the message by the primary MDC, it creates an entry for the new members to its binding table. Finally, the primary MDC sends messages to the other MDCs, which then update their binding tables.

4.3.2. Constructing the Service Zones

This subsection presents the details of the convex hull construction algorithm. Convex hulls are constructed to determine the sensor nodes on the boundary of SZs and form groups. The convex hull construction is based on the Graham scan algorithm [38]. The algorithm first explicitly sorts the nodes in and then applies a linear-time scanning algorithm to finish building the hull. To compute the convex hull , the function CH() performs the following three phases.

Phase I. Select an anchor point (base node) in , normally this is the node with the minimum y-coordinate. In case of a tie, the leftmost node (minimum x-coordinate) in the set is selected.

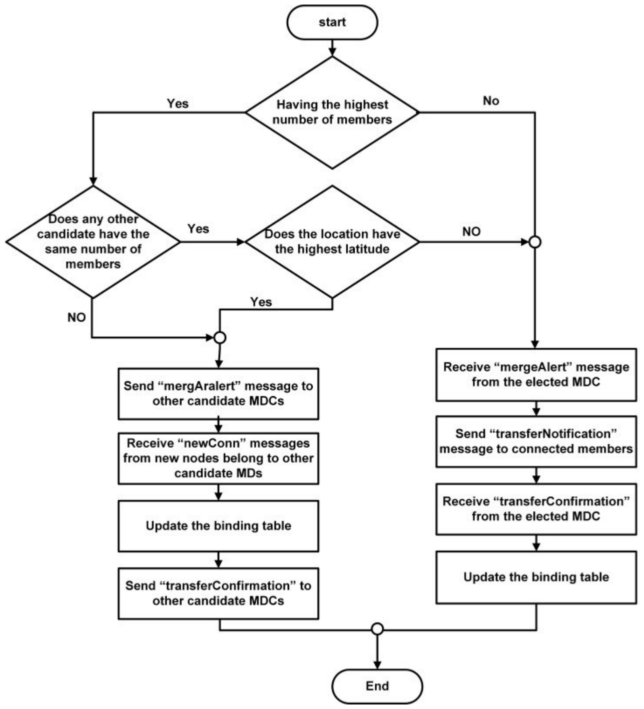
start
Having the highest number of members
Yes
No
Does any other candidate have the same number of members
Yes
Does the location have the highest latitude
NO
Yes
NO
Send "mergAralert" message to other candidate MDCs
Receive "newConn" messages from new nodes belong to other candidate MDs
Update the binding table
Send "transferConfirmation" to other candidate MDCs
Receive "mergeAlert" message from the elected MDC
Send "transferNotification" message to connected members
Receive "transferConfirmation" from the elected MDC
Update the binding table
End

Fig. 8 - Collection zones reconciliation.

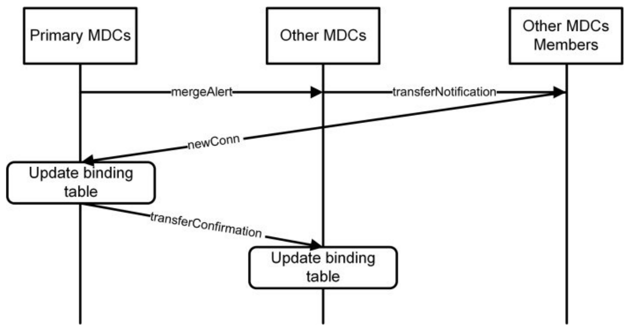

Fig. 9 - CZs reconciliation algorithm messages diagram.

Phase II. Sort the remaining nodes of , i.e., , lexicographically by polar angle, measured in radians. Interior nodes on the ray can not be convex hull points and are excluded during sorting. Once the nodes are sorted, they are connected in counter clockwise order with respect to the anchor node . The result is a simple polygon as shown in Fig. 10. Note that the algorithm performs no explicit computation of angles.

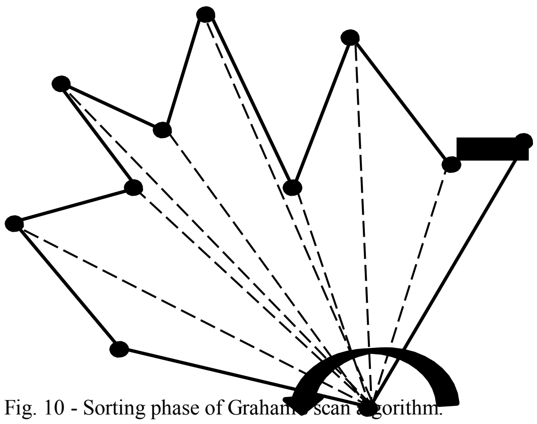

Fig. 10 - Sorting phase of Graham's scan algorithm.

Phase III. After pushing the anchor node onto the stack , nodes are scanned in counter clockwise order, maintaining at each step a stack containing a convex chain surrounding the nodes scanned so far. At each node the following test is performed:

a. If forms a left turn with the last two points in the stack , or if contains fewer than two points, then push onto the stack .

b. Otherwise, pop the last point from the stack and repeat the test for .

The process halts when the algorithm returns to the anchor point , at which point stack stores the vertices of the convex hull of in counter clockwise order. Fig. 11 presents a convex hull after performing Phase III.

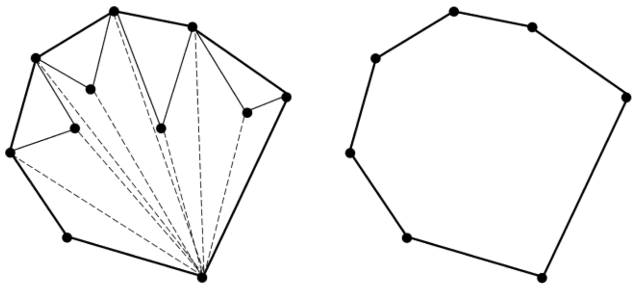

Fig. 11 - phase III of Graham's scan algorithm.

Let the vector represent the line segment between the last two nodes in the stack . To demine that a new node is on the left of the line segment , the MDC applies the right hand-rule, by checking the orientation of the cross product , which is equivalent to equation (7).

Then, the node is left of the line segment if the result of equation is positive . The pseudo-code in Algorithm 2 provides the details of Graham Scan Algorithm for constructing convex hulls.

Algorithm 2: Graham Scan Algorithm

```
Input: a set of points S = {P = (P.x,P.y)}
Begin
Select the rightmost lowest point P0 in S Sort
S radially (ccw) about P0 as a center { Use
isLeft() comparisons
  For ties, discard the closer points
}
Let P[N] be the sorted array of points with P[0]=P0
Push P[0] and P[1] onto a stack Ω
while i < N do
  Let PT1 = the top point on Ω
  If (PT1 == P[0]) then
    Push P[i] onto Ω
    i++
  endif
  Let PT2 = the second top point on Ω
  If (P[i] is strictly left of the line PT2 to PT1) then
    Push P[i] onto Ω
    i++
  else
    Pop the top point PT1 off the stack
  endif
endwhile
END
Output: Ω = the convex hull of S
```

(7)

4.4. Network Clusters Maintenance

The logical CZ and SZ membership requires regular updates. This section provides a complete picture of how network clustering is maintained. For handling changes in network topology due to frequent MDC mobility, the proposed update mechanism is triggered periodically by MDCs. The update mechanism provides a continuous process to keep track of changes in the network.

To reduce the delay in implementing performance-critical logical zone updates, the update mechanism provides local checks and calculations performed by MDCs; sensor nodes only participate in the process when the MDCs detect a change. This mechanism is energy efficient since updates are limited in scope; only the transferring MDCs and interconnected neighboring nodes are aware of the handover.

When an MDC is moving out of its CZ, a new CZ is created and nodes belonging to the corresponding SZ are reconfigured. When an updated CZ crosses its defined SZ boundary, the previously constructed SZ is destroyed and a new SZ will be constructed. An SZ update may remove nodes that are no longer in an MDC vicinity, or add nodes disconnected from another SZ. Fig. 12 shows an MDC moving in a southerly direction and out of its SZ.

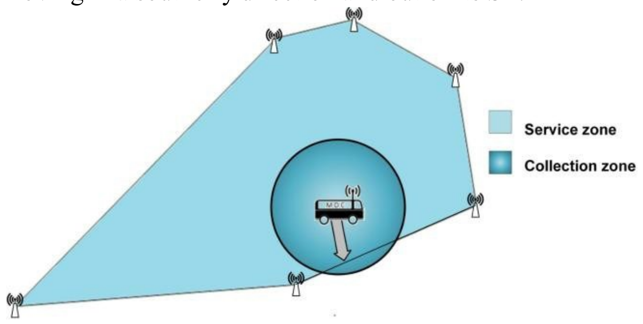

Fig. 12 - Service zone update.

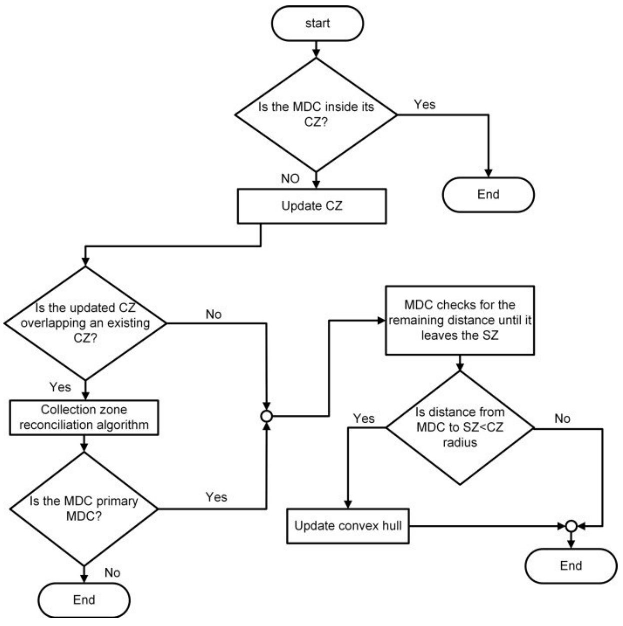
start
Is the MDC inside its CZ?
Yes
End
NO
Update CZ
Is the updated CZ overlapping an existing CZ?
No
Yes
Collection zone reconciliation algorithm
Is the MDC primary MDC?
Yes
No
End
MDC checks for the remaining distance until it leaves the SZ
Is distance from MDC to SZ<CZ radius
Yes
No
Update convex hull
End

Fig. 13 - SZ updating mechanism

The MDC starts by scanning its binding table to determine all nodes that are further than from it (where is the maximum distance of a node to the MDC). These nodes are disconnected by a message. Next, the MDC sends advertisement messages to the new sensor nodes that are within a distance . Upon receiving the advertisement, unconnected sensor nodes respond by a message. The previous procedure excludes nodes that are already connected to the original service area; these nodes only forward the advertisements. Fig. 13 shows the details of the cluster local update mechanism.

The cluster update mechanism is periodically performed by the MDCs. An MDC checks whether it is inside its CZ by comparing the distance between its location and the center of its CZ with the CZ radius. If that distance is greater than the radius of its CZ, then the MDC is not inside its CZ and the CZ will be updated. It is important to point out that the updated CZ could overlap an existing CZ. In this case, the CZ reconciliation algorithm is executed, and hence one SZ would be constructed for both MDCs.

After a CZ update, the MDCs calculate the estimated remaining distance and time in their current SZs. This information is used by the MDC to determine when to update its SZ. Intuitively, the MDC will intersect one of the SZ edges after some certain time. To calculate an estimation for this time and remaining distance for the MDC inside its SZ, the intersection point of the MDC and the SZ edge must be predicted.

Let be the line segment between endpoints, and , the MDC current position and its new location after it crosses the SZ, respectively. The extended line through and is given by the parametric equation (8):

(8)

with the line direction vector. Then the segment contains those points with .

Let a convex hull be given by vertices going counter clockwise around the hull, and let . Also let be the edge (line segment) for ; and be the edge vector. Then, an outward-pointing normal vector for is given by , where is the 2D perpendicular operator.

To determine the hull edge that will intersect with the line segment , we scan the hull edges checking if the vector from to points to the outside of the edge. When , there is no intersection with the edge, so ignore this edge, and continue processing the other edges.

As indicated in Fig. 14, intersection occurs when . =0, since any vector parallel to the edge is perpendicular to the edge normal vector. Substituting for and solving for , we get:

(9)

at the intersection point , is plugged back into the first equation

(10)

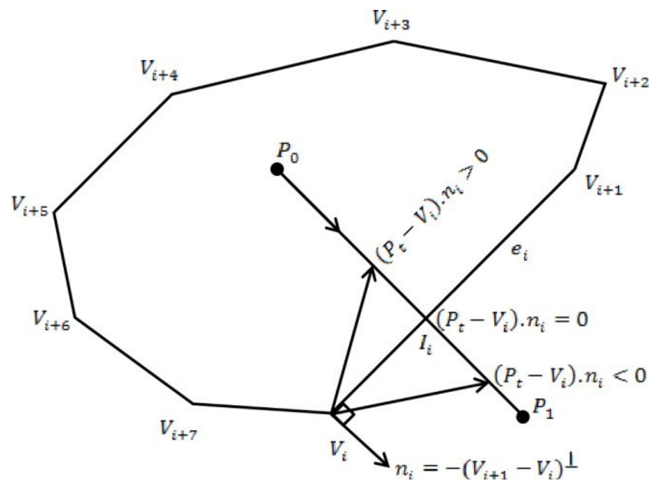

Fig. 14 - MDC and SZ intersection

The pseudo-code in Algorithm 3 provides the details of SZ update prediction algorithm.

Algorithm 3: MDC and SZ intersection

```
Input: a 2D segment S from point   to point
       a 2D convex polygon CH with   vertices

Begin
  if               then   is a single point, so then
    test for point inclusion of    in CH; and
    return the test result (TRUE or FALSE);
  endif
  Initialize:
           for the min intersecting segment  parameter;
                   is the segment direction vector;
  for every                                  do
    Let      an outward normal of the edge   ;
    N = - dot product of ($P_0$-$V_i$) and $n_i$;
    D = dot product of dS and $n_i$;
    if (D == 0) then
      S is parallel to the edge $e_i$
      if (N < 0) then
        $P_0$ is outside the edge $e_i$
        return FALSE since S cannot intersect CH;
      else S cannot leave CH across edge $e_i$ then
        ignore edge $e_i$ and
        continue to process the next edge;
      endif
    endif
    Put t = N / D
  endfor
End
Output:  P(t) = $P_0$ + t * dS
```

5. Handling Orphaned Nodes in DCMDC

This section discusses the orphaned nodes problem and how the DCMDC protocol handles and maintains their connectivity.

Disconnecting nodes during updating service areas can result in 'orphaned nodes'. An orphaned node is a node that is not connected to any MDC. Such a node loses its connectivity to neighboring nodes or is unable to obtain an advertisement message from any of the MDCs as it is located outside the MDCs radio range. An orphaned node may keep attempting to connect to its previous parent.

Orphaned nodes lead to segmentation problems, where the network is divided into many unconnected segments. This situation could also occur when MDCs are located distant from each other, and there are unconnected nodes between the SZs. Such a situation may lead to disconnections and loss of data from orphaned nodes and other parts of the network. To resolve the orphaned node problem, we opted to extend the SZs to the whole monitored area by using the following steps:

1. If a node does not receive an advertisement from an MDC or a gets disconnected, it waits for a back-off interval.
2. If the node still did not receive an advertisement, it uses a reactive discovery approach by sending out an MDC solicitation message to its neighbors to obtain MDC information.
3. Neighbors forward the message to the MDC and wait for reply. In case of the neighbor is also orphaned, the node enters another back off interval to allow their neighbor to obtain the MDC information.
4. The MDC sends its information to the forwarding node.
5. The forwarding node receives the MDC information message and forwards it to the orphaned node.
6. Upon receiving information about the surrounding MDCs, the orphaned node executes the optimal MDC selection algorithm. Choosing the optimal MDC is based on the connection expiration time (CET). In [37], we presented our MDC selection algorithm.
7. Orphaned node chooses the optimal MDC and sends to it a message.
8. The chosen MDC waits for a backoff interval waiting for other messages from other orphaned nodes.
9. The chosen MDC updates its convex hull to join the orphaned nodes.

Unlike the exhausted (or dead) nodes, the orphaned nodes can still receive and transmit messages; thus it is possible to restore them to the network. Handling and minimizing the number of orphaned nodes preserves their energy and reduces signaling overhead, which assists in balancing energy consumption. Connecting orphaned nodes and alleviates network segmentation and energy depletion. Orphaned node join the optimal MDC that keeps them connected for the longest period.

6. Performance Evaluation

The performance of DCMDC was evaluated extensively under diverse conditions and compared against two of its best rivals in the literature, namely, ERMMSDG [25] and MDC/PEQ [26]. These protocols are similar to the

DCMDC in spirit, but different in approach. Both ERMMSDG and MDC/PEQ are mobility management protocols designed for three-tire WSN systems. They use multiple mobile data collectors to collect data from sensor nodes. Furthermore, both protocol use real-life simulation parameters, which resembles the specifications of many existing networks and hardware platforms. Their publications give their full specifications, making it possible for researchers to implement and reproduce the published results. Finally, they achieve best results compared to their contenders in the literature. In this section, we present the simulation parameters, results and analysis.

The simulation scenario consists of nodes randomly placed in the area of . Sensor nodes have wireless radio range of . The transmission and reception power of a sensor node is set to . Four data sources were chosen randomly to generate throughout the simulation. The packet size is for all control messages. Whereas, the size of data packets is set to . Each node is given of initial energy, which is equivalent to energy of two AA batteries. The number of MDCs is set to of the total node population. MDCs were deployed randomly. Their mobility speed reaches up to and they move according to the random waypoint mobility model described in [39, 40]. This travel speed mimics the speed of moving objects in real-life applications, such as wildlife monitoring or battlefield surveillance, where the travel speed of an animal or an armored vehicle is approximately . MDCs wireless radio range can reach up to . The sink node is located at the center of the simulation area and has wireless radio range of . A summary of the simulation parameters and their respective values is shown in Table 2. The chosen simulation parameters for the experiments are based on the iMote2 [41] hardware platform specifications.

Table 2: DCMDC simulation parameters

Parameter	Value
Number of nodes	
Simulation area	
Wireless radio range (SN)	
Wireless radio range (MDC)	
Source nodes data rate	
Number of MDCs	
MDC velocity	
Data packet size	
TX power dissipation	
RX power dissipation	
Mobility Model	Random waypoint

To evaluate the performance of DCMDC, we compare it against the ERMMSDG and the MDC/PEQ protocols through simulation using the NS3 simulator [42]. Their performance is evaluated according to several metrics including: end-to-end delay, packet delivery ratio, packet drop, average energy consumption per node, and network lifetime.

6.1. End-to-End Delay (E2Ed)

E2Ed includes the queuing, transmission, propagation and processing delays. The average delay of all nodes is given by the following equation:

(11)

where is the time a packet is generated, is the time a packet arrives at its final destination and is the number of data packets generated at sensor nodes and received by the sink.

Fig. 15 shows that DCMDC reduces the average end-to-end delay by and compared to ERRMSDG and MDC/PEQ respectively. There are several factors accounting for this outcome. First, the DCMDC algorithm minimizes the packet transmission interruption times and maintains high network connectivity by responding rapidly to any topological changes. On the other hand, MDC/PEQ and ERRMSDG incur larger signaling traffic as a consequence of routing packets through longer paths. Second, nodes in ERRMSDG and MDC/PEQ only consider the signal strength in selecting the serving MDC, while nodes in DCMDC consider the direction, distance and speed of potential MDCs. Thus, DCMDC results in well-delimited dynamic groups of nodes that has less frequent route updates and topology reconfigurations; therefore, reducing potential packet delivery delays. Third, DCMDC predicts the future disconnection time; and hence, nodes use short paths to MDCs that last for longer time. This significantly shortens the propagation and queuing delay.

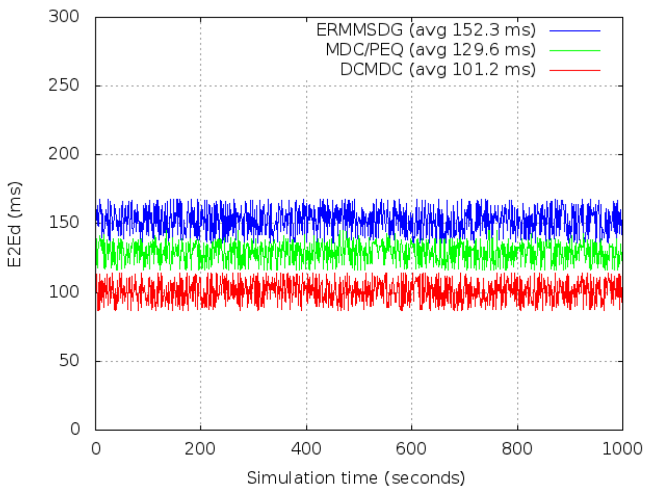

Fig. 15 – Average E2Ed vs simulation time.

6.2. Packet Delivery Ratio (PDR)

PDR is the ratio of packets that are successfully delivered to a destination compared to the number of packets that have been sent by the sender(s). PDR is given as:

(12)

Fig. 16 plots the PDR of the three studied protocols against the simulation time. DCMDC outperforms the ERRMSDG and MDC/PEQ by grouping nodes into smaller SZs and localizing mobility management traffic. The lower mobility management overhead results in smaller number of collisions and reduced data loss due to network congestion. It can be observed that the performance of DCMDC in term of PDR has a frequent fluctuation. It achieves high PDR when the network is stable, i.e., between SZs reconfigurations. At other instances, DCMDC PDR drops below when sensor nodes execute the CZs reconciliation procedure; whereby, the bandwidth utilization increases dramatically due to the heavy exchange of reconfiguration messages. Another reason behind DCMDC's high PDR is the use of the optimal MDC selection scheme, which helps nodes stay connected for a longer time; therefore, increasing the network availability and reducing the dropped packet rate. In ERRMSDG and MDC/PEQ, orphan nodes use direct route to the sink to transfer their data. This leads to conveying data through multiple hops, thereby contributing to an increase of packet collisions and losses. It is also observed that the packet drop increases steadily, when the transmission distances approaches , due to weak signal strength and the travel speed of MDCs.

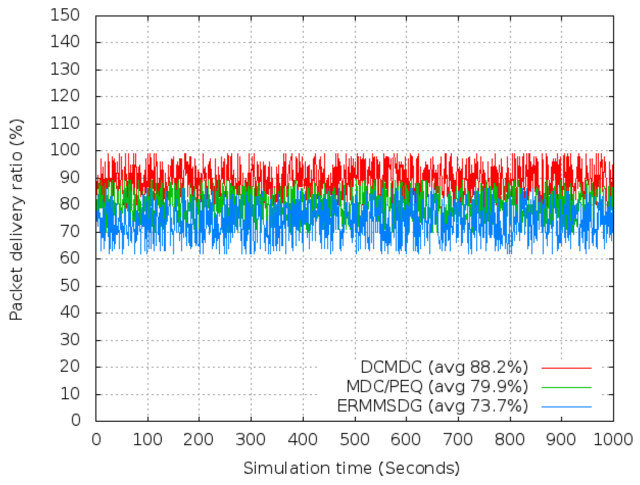

Fig. 16 - PDR vs simulation time.

Fig. 17 shows the PDR of the three studied protocols when varying the number of data sources. DCMDC's PDR drops gradually when increasing the number of data sources, demonstrating DCMDC's ability to handle higher volumes of traffic from different sources. In ERMMSDG, the PDR reaches for the tested scenario. This low delivery ratio is due to the selection of the nearest node to be the relay node, which leads to load-imbalancing. This, combined with the connectivity disconnections due to MDC movement, leads to high packet loss. Furthermore, in situations where an MDC is gathering data from two or more rendezvous zones, the same relay node will be selected for forwarding the traffic. This causes bottlenecks on nodes close to the relay node; consequently, consuming higher bandwidth, and therefore, causing higher packet drop rates. In MDC/PEQ, the PDR reaches for the tested scenario. The high beacon transmission rate as well as relaying data directly to the sink lead to higher bandwidth consumption, and thus higher packet loss, as shown in Fig. 18.

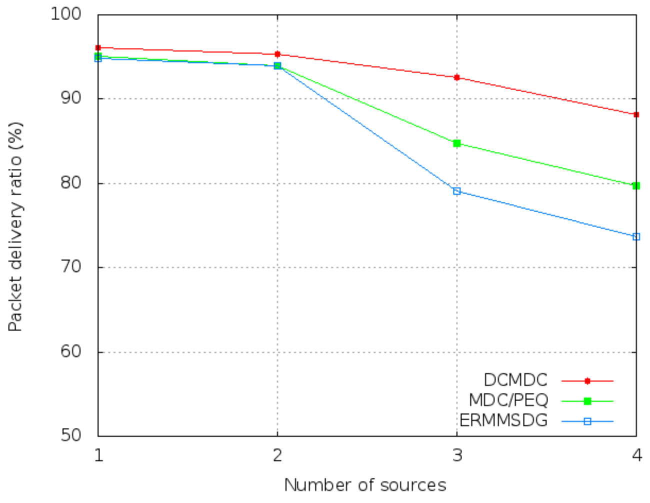

Fig. 17 - PDR vs number of sources.

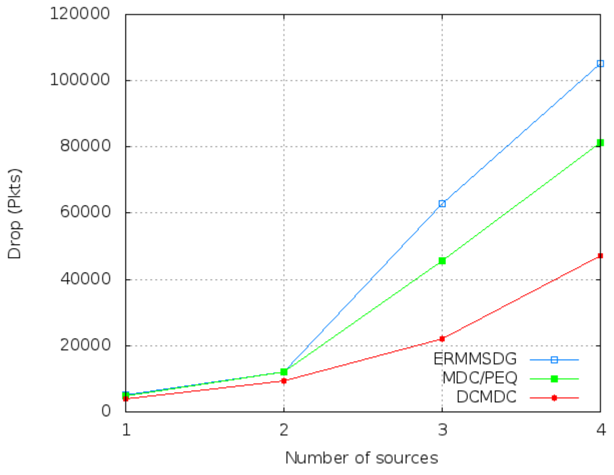

Fig. 18 - Packet drop vs number of sources.

As observed in Fig. 19, the PDR decreases when increasing the speed of the MDCs. As the travel speed of MDCs increases, the probability of errors in data transmission increases. This is because increasing the travel speed of MDCs instigate frequent topological changes that need to be resolved before data collection can be resumed. Consequently, a higher number of packets will be dropped or will arrive late due to buffer overflows and congestion. Yet, DCMDC performed better than ERRMSDG and MDC/PEQ as it isolates the topological updates and limits them to the SZ boundary. In case of DCMDC, the speed of travel is already factored for during the selection of the optimal MDC and nodes are always connected to the MDC with the highest CET.

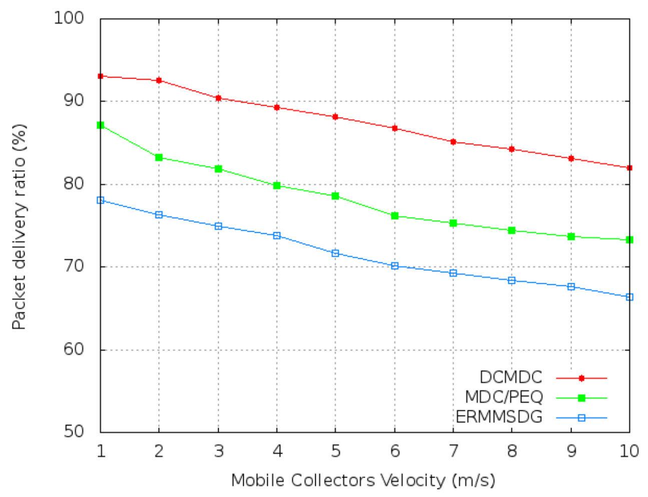

Fig. 19 - PDR vs MDCs velocity.

6.3. *Average Energy Consumption per Node*

Since radio communication is the most power-hungry operation [43], the energy consumption of DCMDC is measured as the cost of mobility management added to the cost of data collection. The average energy consumption of a sensor node is directly related to the operational lifetime of the network.

(13)

where is the initial amount of energy of any sensor node, is the residual energy of the sensor node at the end of the simulation and is the number of sensor nodes in the network.

As shown in Fig. 20, a single MDC network consumes high energy as it causes a signaling ripple effect and results in bottlenecks in areas around itself. The energy consumption decreases gradually when increasing the number of MDCs from to . This is due to distributing the load among the MDCs and the intermediate nodes used to reach them. Furthermore, multiple MDCs can reduce the number of hops that data packets have to traverse. When increasing the number of MDCs to more than 6, the average energy consumption per node starts to increase moderately. This is because when having more MDCs in the network, the number and frequency of SZs updates increase. This results in a gradual rise in the signaling overhead, which dissipates energy gains.

The above findings are on the optimal number of MDC are consistent with the empirical results of [44] (the optimal number of partitions in the network is estimated at about of the total number of nodes in the network). DCMDC performs better than ERMMSDG and MDC/PEQ in terms of energy consumption in all cases. It is capable of reducing energy consumption by and when compared to ERMMSDG and MDC/PEQ respectively. These gains in energy consumption are due to DCMDC's capability of reducing the mobility management overhead and delivering data packets over the shortest route to the MDC, while maintaining load balancing. MDC/PEQ generates high number beacon packets and uses the path to an MDC with the smaller number of hops, which is not always the optimal path in terms of energy consumption. For instance, due to packet loss, MDC/PEQ has to retransmit packets, thus, increasing the energy consumption.

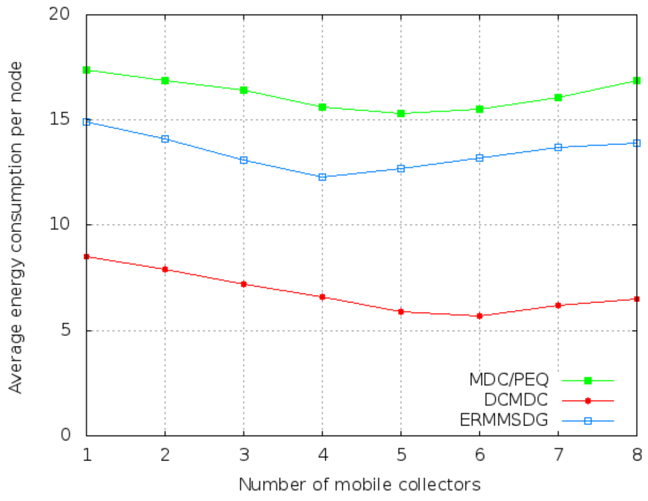

Fig. 20 - Energy consumption vs the number of mobile collectors.

6.4. *Network Lifetime*

Network lifetime is measured as the time duration before the energy level of of the total node population becomes zero. Fig. 21 shows that DCMDC substantially prolongs the network lifetime by and over ERMMSDG and MDC/PEQ respectively. This energy saving is mainly due to nodes joining the MDC offering the longest CET, thus, avoiding frequent handoffs and the costs associated with reestablishing a path to the MDC. In ERMMSDG, relay nodes consume their energy faster than other nodes due to forwarding the data packets from the rendezvous point. Whereas, some MDC/PEQ sensor nodes consume more energy in receiving, processing and forwarding beacons.

Fig. 21 also gives insights into energy balancing in the three studied protocols. When the time interval between the First Node to Die (FND) and the Last Node to Die (LND) decrease, this indicates a more balanced energy consumption among sensor nodes in the network. The time interval between the FND and the LND in DCMDC, ERMMSDG and MDC/PEQ is , and respectively. Thus, DCMDC's energy consumption is and more load balanced compared to ERMMSDG and MDC/PEQ respectively. This is primarily because DCMDC constructs communication links within the SZ with the communication cost as a primary factor. Whereas, MDC/PEQ only relies on the number of hops without considering the link load, quality or reliability.

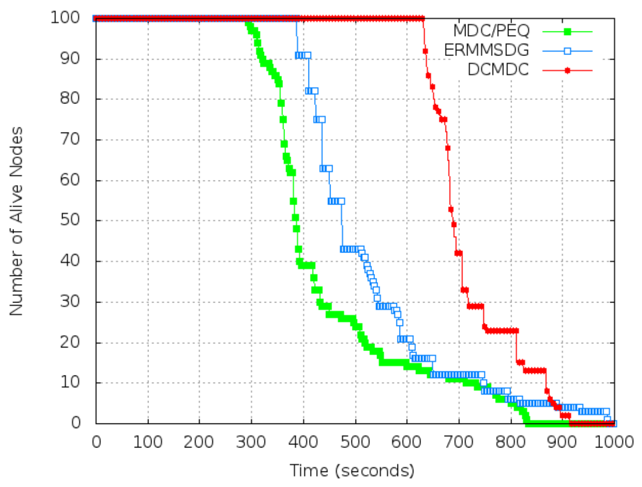

Fig. 21 – Number of alive nodes vs simulation time.

7. Conclusion

This paper addressed the issue of efficient mobility management. Motivated by keeping the data latency and energy consumption to the minimum, a dynamic network clustering technique, called DCMDC, is proposed. Network clustering is a well-tested solution to achieve scalability and load balancing. Grouping nodes into smaller logical sets makes buffer overflows and energy depletion less of a problem. DCMDC manages MDCs mobility and results in a set of well-delimited network clusters of sensor nodes that are updated dynamically. Experimental evaluation showed that DCMDC reduces mobility management cost, end-to-end delay, and energy consumption while increasing the network lifetime and the packet delivery ratio.

There are a number of interesting directions for future work. First, the CZs have the potential to be utilized by data collection approaches that are based on logical grouping of nodes to deliver their intended functionality, e.g., for query scoping or dissemination. Second, more work needs to be done to further reduce the handover interruption time, i.e., the time between disconnecting from the current SZ and connecting to a new one. This can be achieved by developing a precise time prediction algorithm to predict when the SZ needs update. Informing nodes that will be affected with the update process before the time of the update is due gives nodes time to proactively execute the optimal MDC selection algorithm. Consequently, handoffs would be performed more rapidly.

Appendix A. Control Messages of DCMDC Protocol

This appendix presents the control messages format of the proposed DCMDC protocol. In the following, we list the control messages and provide details of their structure:

ID	Type	Location	MembersNo	Distance	HopCount	TimStamp

ID: is the MDC identifier.
Type: is the message type.
Location: is the MDC location information (X position, Y position, track).
MembersNO: is the number of sensor nodes connected to the MDC.
Distance: is the maximum distance that the advertisement message can be forwarded over.
HopCount: is the number of hops that the advMsg has traversed over.
TimeStamp: is the time when the message has been sent.

ID	Type	Location	Energy	TimeStamp

ID: is the MDC identifier.
Type: is the message type.
Location: is the sensor node location information(X position , Y position, track).
Energy: is the residual every of the sensor node.
TimeStamp: is the time when the message has been sent.

ID	Type	Location	Energy	TimeStamp

ID: is the MDC identifier.
Type: is the message type.
Location: is the MDC location information(X position, Y position, track).
MembersNo: is the number of sensor nodes connected to the MDC.
TimeStamp: is the time when the message has been sent.

ID	Type	TimeStamp

ID: is the primary MDC identifier.
Type: is the message type.
TimeStamp: is the time when the message has been sent.

ID	Type	nMDC_ID	nMDC_Location	TimeStamp

ID: is the old MDC identifier.
Type: is the message type.
nMDC_ID: is the new MDC identifier.
nMDC_Location:is the new MDC location information(X position, Y position, track).
TimeStamp: is the time when the message has been sent.

ID	Type	tMembersNo	TimeStamp

ID: is the MDC identifier.
Type: is the message type.
tMembersNo: is the number of the transferred sensor nodes.
TimeStamp: is the time when the message has been sent.

References

[1] A. Abuarqoub, "Cooperative Mobility Maintenance Techniques for Information Extraction from Mobile Wireless Sensor Networks," PhD, School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester Metropolitan University's Research Repository, 2014.

[2] G. Mao, *et al.*, "Wireless sensor network localization techniques," *Computer Networks,* vol. 51, pp. 2529-2553, 2007.

[3] A. Savvides, *et al.*, "Dynamic fine-grained localization in Ad-Hoc networks of sensors," presented at the Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy, 2001.

[4] N. Bulusu, *et al.*, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications,* vol. 7, pp. 28-34, 2000.

[5] H. Liu, *et al.*, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 37, pp. 1067-1080, 2007.

[6] M. Femminella and G. Reali, "Low Satellite Visibility Areas: Extension of the GPS Capabilities to Deploy Location-Based Services," *IEEE Vehicular Technology Magazine,* vol. 7, pp. 55-65, 2012.

[7] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer Communications,* vol. 30, pp. 2826-2841, 2007.

[8] S. Soro and W. B. Heinzelman, "Cluster head election techniques for coverage preservation in wireless sensor networks," *Ad Hoc Netw.,* vol. 7, pp. 955-972, 2009.

[9] K. A. Darabkh, *et al.*, "Performance evaluation of selective and adaptive heads clustering algorithms over wireless sensor networks," *Journal of Network and Computer Applications,* vol. 35, pp. 2068-2080, 2012.

[10] Q. Dong and W. Dargie, "A Survey on Mobility and Mobility-Aware MAC Protocols in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials,* vol. 15, pp. 88-100, 2013.

[11] M. Hammoudeh, *et al.*, "Map as a Service: A Framework for Visualising and Maximising Information Return from Multi-ModalWireless Sensor Networks," *Sensors* pp. 22970-23003, 2015.

[12] J. Y. Yu and P. H. J. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials,* vol. 7, pp. 32-48, 2005.

[13] Y. Zhang, *et al.*, *RFID and Sensor Networks: Architectures, Protocols, Security, and Integrations*: CRC Press, Inc., 2009.

[14] N. Blefari-Melazzi, *et al.*, "Autonomic control and personalization of a wireless access network," *Computer Networks,* vol. 51, pp. 2645-2676, 2007.

[15] S. Jabbar, *et al.*, "Multilayer cluster designing algorithm for lifetime improvement of wireless sensor networks," *J. Supercomput.,* vol. 70, pp. 104-132, 2014.

[16] Ko, *et al.*, "Controlled Sink Mobility Algorithms for Wireless Sensor Networks," *International Journal of Distributed Sensor Networks,* vol. 2014, p. 12, 2014.

[17] Y. Sheng, *et al.*, "Routing protocols for wireless sensor networks with mobile sinks: a survey," *Communications Magazine, IEEE,* vol. 52, pp. 150-157, 2014.

[18] D. Das, *et al.*, "Multiple-sink placement strategies in wireless sensor networks," in *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, 2013, pp. 1-7.

[19] B.Sudhakar and K.Sangeetha, "Multi Sink based Data Collection Scheme for Wireless Sensor Networks " *International Journal of Innovative Research in Computer and Communication Engineering* vol. Vol.2, pp. 1139-1146, March 2014 2014.

[20] D. Amine, *et al.*, "Energy Efficient and Safe Weighted Clustering Algorithm for Mobile Wireless Sensor Networks," *Procedia Computer Science,* vol. 34, pp. 63-70, 2014/01/01 2014.

[21] L. Peng and J.-b. Xu, "ECDGA: An Energy-Efficient Cluster-Based Data Gathering Algorithm for Mobile Wireless Sensor Networks," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, 2009, pp. 1-4.

[22] J. G. Olascuaga-Cabrera, *et al.*, "Self-organization of mobile devices networks," in *System of Systems Engineering, 2009. SoSE 2009. IEEE International Conference on*, 2009, pp. 1-6.

[23] G. S. Sara, *et al.*, "Energy Efficient Clustering and Routing in Mobile Wireless Sensor Network," *International Journal of Wireless & Mobile Networks (IJWMN),* vol. 2, pp. 106-114, November 2010 2010.

[24] T. Ducrocq, *et al.*, "Energy-based clustering for wireless sensor network lifetime optimization," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, 2013, pp. 968-973.

[25] P. Madhumathy and D. Sivakumar, "Enabling energy efficient sensory data collection using multiple mobile sink," *China Communications,* vol. 11, pp. 29-37, 2014.

[26] R. W. N. Pazzi and A. Boukerche, "Mobile data collector strategy for delay-sensitive applications over wireless sensor networks," *Comput. Commun.,* vol. 31, pp. 1028-1039, 2008.

[27] T. Alsboui, *et al.*, "Information Extraction from Wireless Sensor Networks: System and Approaches," *Sensors & Transducers Journal,* vol. 14-2, pp. 1-17, March 2012.

[28] M. Hammoudeh, *et al.*, "An Approach to Data Extraction and Visualisation for Wireless Sensor Networks," in *Networks, 2009. ICN '09. Eighth International Conference on*, 2009, pp. 156-161.

[29] J. Liu and Y. Hu, "A balanced and energy-efficient clustering algorithm for heterogeneous wireless sensor networks," in *Wireless Communications and Signal Processing (WCSP), 2014 Sixth International Conference on*, 2014, pp. 1-6.

[30] D. Xie, *et al.*, "Multiple mobile sinks data dissemination mechanism for large scale Wireless Sensor Network," *China Communications,* vol. 11, pp. 1-8, 2014.

[31] M. Zhao, *et al.*, "Mobile Data Gathering with Load Balanced Clustering and Dual Data Uploading in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing,* vol. 14, pp. 770-785, 2015.

[32] O. Saukh, *et al.*, "Convex groups for self-organizing multi-sink wireless sensor networks," in *Industrial Electronics, 2009. IECON '09. 35th Annual Conference of IEEE*, 2009, pp. 2624-2629.

[33] A. Shahid, *et al.*, "Proactive multipath data dissemination for Multimedia Sensor Networks," in *Multitopic Conference (INMIC), 2012 15th International*, 2012, pp. 349-354.

[34] J. Niu, *et al.*, "R3E: Reliable Reactive Routing Enhancement for Wireless Sensor Networks," *Industrial Informatics, IEEE Transactions on,* vol. 10, pp. 784-794, 2014.

[35] G. S. Sara, *et al.*, "Energy Efficient Mobile Wireless Sensor Network Routing Protocol," in *Recent Trends in Networks and Communications: International Conferences, NeCoM 2010, WiMoN 2010, WeST 2010, Chennai, India, July 23-25, 2010. Proceedings*, N. Meghanathan, *et al.*, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 642-650.

[36] C. Hsung-Pin and H. Shun-Chih, "A hybrid intelligent protocol in sink-oriented wireless sensor networks," in *Information Security and Intelligence Control (ISIC), 2012 International Conference on*, 2012, pp. 57-60.

[37] Omar Aldabbas, *et al.*, "Unmanned Ground Vehicle for Data Collection in Wireless Sensor Networks: Mobility-aware Sink Selection," *The Open Automation and Control Systems Journal,* vol. 8, pp. 35-46, 2016.

[38] R. L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set," *Information Processing Letters,* vol. 1, pp. 132-133, 1972.

[39] J. Broch, *et al.*, "A performance comparison of multi-hop wireless ad hoc network routing protocols," presented at the Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, Dallas, Texas, USA, 1998.

[40] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*. vol. 353, T. Imielinski and H. Korth, Eds., ed: Springer US, 1996, pp. 153-181.
[41] Intel, "Intel Mote 2," in *Engineering Platform Data Sheet* ed, 2006, pp. 1-9.
[42] nsnam. (2011, Retrieved 18 April 2012). *NS-3*. Available: from http://www.nsnam.org/
[43] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM,* vol. 43, pp. 51-58, 2000.
[44] W. R. Heinzelman*, et al.*, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," presented at the Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 - Volume 8, 2000.

Abdelrahman Abuarqoub is an Assistant Professor in Computer Science, Head of Department of Computer Science, and Vice Dean of the Faculty of Information Technology at the Middle East University of Jordan. He received his PhD in Computer Science from the Manchester Metropolitan University in 2014, his MSc (Distinction) in Data Telecommunications and Networks from the University of Salford in 2011, his BSc in Computer Networks Systems from Applied Science University/Jordan in 2009. His research focuses on Wireless Sensor Networks, ubiquitous and mobile computing, specifically in Internet of Things.

Mohammad Hammoudeh is a Senior Lecturer in Computer Networks and Security in the School of Computing, Mathematics and Digital Technology at the Manchester Metropolitan University. He received his Ph.D. in Computer Science from the University of Wolverhampton in 2009, his MSc in Advanced Distributed Systems from the University of Leicester in 2007 and his BSc (Hons) in Computer Communications from the Arts, Sciences & Technology University in Lebanon in 2004. He is the co-founder and member of the FUture Networks and Distributed Systems research Group (FUNDS). He is the founder and head of the MMU IoT Lab.

Bamidele Adebisi received his Master's degree in advanced mobile communication engineering and Ph.D. in communication systems from Lancaster University, UK. Before that, he obtained a Bachelor's degree in electrical engineering from Ahmadu Bello University, Zaria, Nigeria. He was a senior research associate in the School of Computing and Communication, Lancaster University between 2005 and 2012. He joined Metropolitan University, Manchester in 2012 where he is currently a Reader in Electrical and Electronic Engineering. He has worked on several commercial and government projects focusing on various aspects of wireline and wireless communications. He is a member of IET and a senior member of IEEE.

Sohail Jabbar is a Post-Doctorate Researcher at Network Lab, Kyungpook National University, Daegu, South Korea. He has been Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology (CIIT), Sahiwal and headed Networks and Communication Research Group at CIIT, Sahiwal. He received many awards and honors from Higher Education Commission of Pakistan, Bahria University, CIIT, and the Korean Government. He received the Research Productivity Award from CIIT in 2014 and 2015. He has been engaged in many National and International Level Projects.

Ahcene Bounceur is an associate professor of Computer Science at the university of Brest (UBO). He is a member of the Lab-STICC Laboratory (MOCS Group). He received a Ph.D. in Micro and nano-electronics at Grenoble INP, France in 2007. He received the M.S. degrees in Operations Research from ENSIMAG, Grenoble, France in 2003. From April 2007 to August 2008, he was a postdoctoral fellow at TIMA Laboratory. From September 2007 to August 2008, he was with Grenoble INP, France where he was a temporary professor. He has obtained the 3rd place of the Annual IEEE Test Technology Technical Council (TTTC-IEEE) Doctoral Thesis Contest, VLSI Test Symposium, Berkeley, USA, May 2007. His current research activities are focused on: Tools for physical simulation of Wireless Sensor Networks (WSN), parallel models for accelerating simulations and predicting parameters in WSN, sampling methods for data mining, development of CAT (Computer Aided Test) tools and statistical modeling of analog, mixed-signal and RF circuits. He is the coordinator of the project ANR PERSEPTEUR and a partner of the project Suidia.

Hashem S. Al-Bashar is a Computer Science student and a research associate at the Middle East University of Jordan. His current research interests lie in the areas of wireless sensor networks, data science, internet of things, big data, networking, cloud computing, programming and data warehousing.