J. Parallel Distrib. Comput. 109 (2017) 245–257

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Efficient disjoint boundary detection algorithm for surveillance capable WSNs

Shailendra Shukla [a,*], Rajiv Misra [b], Animesh Prasad [b]

[a] Department of Computer Science and Engineering, Jaypee University of Information Technology, Shimla, India
[b] Department of Computer Science and Engineering, Indian Institute of Technology, Patna, India

HIGHLIGHTS

- Energy efficient boundary detection algorithm for monitoring systems is proposed.
- Boundary detection algorithm is modeled for quasi unit disk graph (QUDG).
- To ensure coverage and reduction in redundant nodes, a packing technique is proposed.
- Maximizing the lifetime of boundary node is formulated as maximum domatic partition.

ARTICLE INFO

Article history:
Received 25 April 2016
Received in revised form 27 April 2017
Accepted 1 June 2017
Available online 21 June 2017

Keywords:
Distributed sensor networks
Pervasive computing
Surveillance

ABSTRACT

Wireless Sensor Networks (WSN) have given a new approach for applications such as surveillance, tracking, and monitoring. In such monitoring systems, the sensor nodes at the network boundary require remaining vigilant to detect events like objects entering and leaving the area under surveillance. Nodes performing surveillance of target area can be classified as boundary node and others as an interior node. Due to continuous vigilance, boundary node suffers from quicker energy exhaustion (compare to internal nodes) which results to a shorter life span of the node. None of the reported works on boundary detection algorithms have considered the problem of maximizing the lifetime of boundary nodes. The problem of maximizing the lifetime of boundary nodes collectively addresses the problem of an optimal number of boundary nodes detection and maximization of boundary node's lifetime. In this work, we formulate the problem of the maximum lifetime of the boundary as the problem of partitioning the boundary region of the network into disjoint dominating sets of a boundary region with the aim of maximizing the number of disjoint dominating sets of a boundary region in the network. This is the problem of the domatic partition of boundary region in graph theory, where each partition is dominating the set of boundary region (i.e., not the dominating set of entire network). Rotating the disjoint dominating sets of boundary region in the network resulting into maximizing the lifetimes of boundary nodes. We found that our disjoint dominating sets of boundary region algorithm have an approximation ratio $((4.8+\ln 5)opt+1.2)$, where *opt* is the minimum size dominating set which has the same approximation ratio as the minimum dominating set problem. The time complexity of boundary detection algorithm is $\mathcal{O}(mn^2)$ where $m=\left|E_u^{k\backslash 1}\right|$, $n=\left|N_u^{k\backslash 1}\right|$ and k is the radius of graph. Simulation results reveal that using our proposed domatic partition of boundary region algorithm, the boundary node saves 35.71% energy lesser as compared to without domatic partition based boundary detection algorithm. To support our claim, we extended our work one step further and performed the experimental analysis of our proposed algorithm on *telosb* motes.

1. Introduction

Wireless Sensor Networks (WSN [1]) have given a new approach for applications such as surveillance, tracking, and monitoring. Such a monitoring system requires detecting events like objects entering and leaving the area under surveillance. Wireless sensor networks (WSN) are built upon tiny low-cost nodes with

* Corresponding author.

E-mail addresses: shailendra.shukla@juit.ac.in (S. Shukla), rajivm@iitp.ac.in (R. Misra), animeshprasad3@gmail.com (A. Prasad).

http://dx.doi.org/10.1016/j.jpdc.2017.06.002
0743-7315/

severe resource constraints, often deployed randomly in the hostile environment which remains inaccessible for humans to recharge its battery source. Energy-efficiency poses a challenge for designers of algorithms and protocols in WSN. The energy constraints and high cost prevent to use GPS devices for location awareness [10,17].

Nodes performing surveillance of target area can broadly be classified based on their roles into interior nodes and boundary nodes. Nodes at the boundary of target area perform surveillance and monitor event enclosed within the WSN [30]. Topological holes [7,13] are formed in WSN dynamically due to malfunctioning (or failure) of sensor nodes. Dynamic nature of nodes (mobility) in topology adds another challenge to the boundary detection problem in WSN. The reported works on boundary problem [6,9] either ignore the presence of holes in topology or assume the knowledge of location using GPS [8,30]. Identifying the boundary of the sensor network using only the topological information can yield realistic results, but fails miserably when the networks are highly sparse [10]. Access to limited computational resources (i.e., small processing power and unavailability of global location information) prevents the use of complex mathematical and statistical computations based approaches for boundary detection [7, 14]. The algorithms based on network-wide flooding [31] face major problem in handling flooding and synchronization. In 3D boundary detection scheme, the authors have used the concepts of triangulation using anchors similar to prior reported works [2] to avoid localization [5,20,26,27]. The drawback of 3D boundary scheme [33] is their assumption of initial anchors (or bootstrap nodes) for initialization.

In surveillance applications, the boundary nodes need to remain active always to perform monitoring of the target area, whereas the interior nodes can sleep for energy savings resulting in an energy depletion boundary node faster compared to interior nodes in homogeneous WSN. There is no work reported in the literature to the best of our knowledge which addresses the lifetime of boundary node problem in WSN. A naive approach to run any existing boundary detection algorithm periodically to identify fresh boundary nodes will lead to high overhead, so extending the lifetime of boundary node problem needs to be addressed at the algorithm design level. The existing boundary detection algorithm cannot be modified (or adopted) effectively for lifetime problem of boundary nodes. In this work, we have proposed a distributed algorithm for detecting the disjoint boundaries once and switch the boundary to extend the lifetime of boundary nodes and WSN.

Maximizing the lifetime of the boundary can be modeled as domatic partitioning of the boundary region. The dominating set of the boundary region results in the formation of a single layer of the boundary. Domatic partitioning boundary region using above formed dominating sets forms multiple boundaries by assigning substitutes for each node in each boundary node communication range disk neighborhood and thus decreases the duty cycle. Using Ore's theorem, the boundary region contains at least two disjoint boundary sets. Hence, rotating boundary between these boundary sets increases the lifetime of the main boundary.

The remainder of this paper is organized as follows. We have briefly summarized related work in Section 2. Sections 3 and 4 present the system model, problem formulation, and contribution of our work. Section 5 covers our proposed disjoint boundary detection algorithm. Sections 6 and 7 cover the simulation result and experimental work & results and finally we conclude our work in Section 8.

2. Related work

The reported works on boundary detection algorithms [16] are classified on the basis of their approaches as (a) geometrical approach, (b) statistical approach and (c) topological approach. The geometrical approach is based on the prior knowledge of nodes location in networks. The algorithms based on statistical approach use complex mathematics and node distribution in the network. In [10,11], Fekete et al. proposed a distributed algorithm where nodes are deployed uniformly without using location information, but require connectivity degree of 100 or higher which is practically unfeasible. A Coverage and hole detection based algorithm is proposed in [13] and [14]. Both [13] and [14] assumes high density, randomly deployed large networks with limited communication range, which becomes impractical to have an average node degree as 100 or higher. Extensive computations in the statistical approach make boundary detection impractical in resource constraint sensor networks.

The topological approach uses topological information such as combinatorial structure like flower and argumental cycle in the networks [17] but fails when the networks are highly sparse [10]. Flooding-based algorithms [31]use cut nodes for flooding the networks and involve three cycles of flooding for outer boundary node detection. In [12], they use iso-counter techniques for the hole boundary detection and network boundary detection, which iterates again requiring flooding network-wide with three different seed nodes. Prior work [4] uses geometric methods for boundary node detection and it assumes that node knows the distance between neighbors and topology is unaffected by communication holes. In [9], the author proposes a boundary identification algorithm which uses heuristics localized convex hull. Recent works have used the concepts of triangulation, landmarks and anchor nodes. The drawback of 3D boundary detection algorithm [33] is its assumption of initial anchors for initialization.

Other reported work of boundary node detection is on issues like energy hole problem where sink node disconnects itself from the remaining nodes that have plenty of energy, due to energy depletion at nodes near the sink node; such scenario creates a void region problem. Reported papers [28,32] have proposed the solution to mitigate the energy hole issue. They have used the theoretical and stochastic node deployment (using similarity/betweenness centrality) based approaches to counter it. Another application aspect of boundary node/virtual coordinate systems is their use in geographic greedy routing. Geographic greedy routing has the disadvantage of communication void region problem (i.e. to handle the local minimum situations) [22–25]. To counter it, papers [24] and [23,25] propose distributed and scalable novel approaches (Curved Stick (CS) [24] and novel geographic routing [23,25]). The proposed approach only depends on the local nature and does not retain memories (i.e., only localization [24] and RSSI values are required).

To the best of our knowledge, none of the reported boundary detection algorithms addresses the lifetime problem of boundary nodes.

3. System model

A collection of sensor nodes deployed over a plane geometric region L can be represented in the form of graph $G = (V, E)$. Here, vertices $V = \{v_1, v_2, \ldots, v_n\}$ represent the sensor nodes in the network and E represents the edge between nodes. The neighborhood of a node $v \in V$ is denoted by Nb_v. Each node can sense the occurrence of events in its sensing region r. The union of the sensing region of boundary nodes is referred to as sensing region (r_{net}) which covers the monitored region. Sensing region (r_{net}) monitors the network, which is typically and significantly larger than the sensing range of a single node($r < r_{net} < T_r$) but less than the communication range (T_r) of network. Each node communicates with its adjacent nodes if they are in proximity, i.e., within the range of r. Throughout the work, we have assumed that node coordinates are unavailable and nodes cannot determine their orientations with other nodes.

4. Problem formulation and contribution

Each node can sense the occurrence of events in its sensing region r. Union of node's sensing region is referred to as sensing region (r_{net}) of L. During surveillance, any event occurs inside the sensing region (r_{net}) of a node is tracked. Throughout the work, we assume that nodes communicate with its adjacent nodes if they are in proximity (i.e., within the range of r) and node coordinates are unavailable.

4.1. Definitions

Definition 1 (*Interior Node (t)*)**.** A interior node $t \in V$ is a node that is enclosed in the closed region bounded with at least three nodes.

Definition 2 (*Boundary Nodes (BN)*)**.** Nodes that lie exactly at the periphery band of plane geometric region L are called boundary nodes. Two neighbor nodes v_i and v_j are defined as boundary nodes when at least one of their intersection points cannot be covered by their direct neighbors (excluding nodes v_i and v_j) or a node $u \in V$ is said to be a boundary node if u is not an interior node.

Definition 3 (*Network Boundary (Ntb)*)**.** Path forming closed cycle of the boundary nodes of the sensor network is defined as a network boundary. The network boundary is an application dependent abstraction such as an edge connected boundary or the sensing range connected boundary.

Definition 4 (*Boundary Region (BR)*)**.** The coverage region of boundary nodes is defined as boundary region.

Definition 5 (*Dominating Set (D_{BR}) of Boundary Region*)**.** A dominating set for a boundary region (say graph BR = (V', E')) is a subset (D_{BR}) of boundary nodes such that every boundary node not in (D_{BR}) is adjacent to at least one member of (D_{BR}).

Definition 6 (*Domatic Partition (DP_{BR}) of Boundary Region*)**.** A domatic partition of boundary region is a partition of *BR* into disjoint sets ($D_{BR_1}, D_{BR_2}, \ldots, D_{BR_k}$) such that each D_{BR_i} is a dominating set of boundary region and maximum value of k is domatic number.

Definition 7 (*Connectedness*)**.** Nodes v_i and v_j are said to be connected if there exists a edge sequence $\{v_1, v_2, \ldots, v_m\}$ such that $\{v_i, v_{i+1}\} \in E$ for $i = \{1, 2, \ldots, m-1\}$.

4.2. Problem formulation

The problem formulation of this work is identifying and partitioning the boundary nodes into a maximum number of disjoint sets of the boundary where each boundary set is a dominating set of boundary region covering the network boundary. The maximum lifetime of boundary node problem in the sensor network can be modeled as maximum size domatic partition of boundary nodes, where each partition is a dominating set of the boundary region.

4.3. Contribution

- Proposed a connectivity based boundary detection algorithm for a quasi unit disk graph (*QUDG*) model.
- To ensure coverage and reduction in redundant nodes, a packing technique is introduced to prune boundary region.
- Maximum lifetime of boundary nodes problem is formulated as the maximum domatic partition of boundary nodes such that partition is a dominating set of boundary node. Duty cycling through the dominating set of boundary nodes maximizes the lifetime of boundary nodes.

5. DBN: Disjoint boundary nodes detection algorithm

Boundary detection algorithm for QUDG in WSN

Our algorithm is modeled for Quasi Unit Disk Graph (QUDG) connectivity model, where two nodes are connected if the euclidean distance between the nodes are equal to or less than d ($0 < d < 1$). If the distance between the nodes is greater than 1, then no link exists between the nodes; if the range is in between d and 1, then the link may or may not be available.

Practically, in WSN, signal strength is strong up to certain distance (parameter d) which ensures the communication but beyond that reception becomes impossible due to physical limitation or signal weakness (when reaches to parameter 1). If graph $G : (V, E)$ is embedded as straight line P: $V \rightarrow R^2$ of networks where line segment corresponds to communication link and end points as vertex of network. The embedding model P in d-quasi unit disk embedding (d-QUDE where $d \leqslant 1$) is defined as follows:

$$(v_i, v_j) \in E \Rightarrow \left\|P(v_i) - P(v_j)\right\|_2 \leqslant 1 \tag{1}$$

and

$$\left\|P(v_i) - P(v_j)\right\|_2 \leqslant d \Rightarrow (v_i, v_j) \in E. \tag{2}$$

When $d = 1$ then 1-*QUDG* is called unit disk graph connectivity model and *d-QUDG* is called d-quasi unit disk graph. Our connectivity model in this work is *d-QUDG*, the smaller the value of d (discussed in reported work [17]) is, the more general and relevant the communication model will be.

5.1. Working of connectivity-based boundary detection algorithm in QUDG

This algorithm uses the separability property of QUDG [3,19] for boundary node detection. Network separability [3] property makes the network algorithm more efficient and increases their computation time. The separator of a graph G is defined as the set of the graph which divides the graph into two subgraphs which stores localized property. To achieve separability property, the algorithm constructs a QUDG network using grid graph (H). The grid graph (H) is a sparse version of QUDG which preserves the representation of the QUDG. As a result, the connectivity between the nodes can easily be maintained between the QUDG and grid graph. To make the graph more separable, an auxiliary graph T [3] is used over the grid graph H in which a larger graph is imposed on grid graph H to make it planar. Virtual vertices are added in the middle of the diagonal edge to eliminate all the middle edges. All the mapped virtual vertex in T is denoted by the red vertex and all the other nodes are represented by the black vertex. Red vertex is given 0 weight and black were given weight equal to the number of the vertex in H. Finally, any arbitrator node (v_i) initiates modified breadth first search algorithm (MBFS) to detect circle in T. MBFS is executed for $k = 2$ to $2 \leqslant k \leqslant rad(G)$ where k is neighbor k-hop away from node v_i. Graph induced by all the k-hop away neighbors $v_k(v_i)$ is $G_{vu}k \setminus 1$ where $2 \leqslant k \leqslant rad(G)$. Boundary detection algorithm in *QUDG* is discussed in algorithm 1.

Algorithm 1 Connectivity based boundary detection algorithm for *QUDG* in *WSN*

1: Construct a grid of cell size $\frac{r}{\sqrt{2}\times\frac{r}{\sqrt{2}}}$ and impose it on the QUDG graph (parameter R and r). Each cell of grid graph corresponds to one vertex of G. Let the grid graph $G_H = (V_H, E_H)$ be the corresponding vertex positions on the cell. And edge between the two vertices of cell in H exists if at least one edge connects the two vertices of G_2 in two corresponding cell.
2: Construct a auxiliary graph T on the obtained grid graph (impose another grid of size $(R + \frac{r}{\sqrt{2}}) \times (R + \frac{r}{\sqrt{2}})$ on grid graph $(V_H \times E_H)$.
3: Each cell in the auxiliary graph T has one black vertex which is placed at the center of the cell. Each black vertex is assigned with the weight which is equal to the number of nodes in the cell (i.e., the number of vertices in corresponding grid graph H).
4: Two black vertices $(u, v) \in T$ of the corresponding cell are considered to be connected if at least one edge is connecting the two vertices of two different cells.
5: Add red vertex at each intersection of two edges formed by 4 black vertices and replace it with 4 edges connecting the red node with other four black vertices with four edges. Red vertex is always assigned with weight 0.
6: Once the separable graph T is formed, any arbitrary node (v_i) initiates the algorithm and chooses a randomly node from its neighbor Nb_{v_i} (let us say v_j). Node v_j executes the modified breath first search *MBFS* till it reaches cycle. If the v_j forms the cycle then the node v_i is interior node else compute the search k hop neighbor for $k = 2$ to $rad(G)$ (where $rad(G)$ is radius of graph which is always $\geq k$) till it forms cycle else v_i is boundary node.

5.2. Correctness of disjoint boundary node detection algorithm

The number of boundary nodes detected by algorithm 1 is not optimal. At certain areas, the boundary nodes are densely aggregated while at certain areas boundary nodes are highly sparse and may even cause dis-connectivity in the boundary (message from a boundary node cannot reach to another by or come back to itself using only boundary nodes). For getting an optimal number of boundary nodes at the same time ensuring connectivity, we need to add some interior nodes to the previously identified boundary to get network boundary. Therefore, each boundary node identifies additional nodes on the shortest path to the nearest boundary as connector nodes (*CN*). The boundary node together with the connector nodes forms a circuit (connected graph). The next step in optimizing the number of nodes is packing of half communication radius $(\frac{R}{2})$ balls on the boundary region. Every node in a ball of the packing dominates all other nodes that belong to the same ball. To minimize the number of balls, overlap among the balls should be minimized, but it could not be reduced to zero because in the case two farthest points of the ball would be $2R$ apart. Keeping these points in mind, we propose the following technique for packing (Packing and Coloring Technique 1).

- All the boundary nodes flood the network to know their nearest boundary nodes.
- Each boundary node runs the shortest path algorithm to its nearest neighbors to get two disjoint shortest paths.
- The resulting nodes along the shortest path mark themselves as connector nodes.
- Consider this close circuit formed by boundary nodes and the connector nodes and run leader election algorithm in the ring based on minimum connectivity in half communication range as a key. Break tie arbitrarily (or higher *node id*).
- Leader node starts packing by picking a boundary or connector node furthest in its half communication range. The packing/fitting of balls start with the leader node having just itself as the center of the first ball and then it selects nodes for increasing the size of the packing. Packing can be increased only in one direction or in both directions simultaneously.
- After selecting the next ball center, the node performs coloring of the colorless nodes in its half communication range. Coloring can be performed randomly or by using the nearest node first technique.
- A node is added to packing as the center of the ball only if it is colorless. Hence, an intermediate node only adds single node (in a single direction) to packing. The termination condition, thus is the inability to find any colorless node to be added to packing which occurs when two end points of a packing meet.
- Same color nodes from each ball form a boundary.

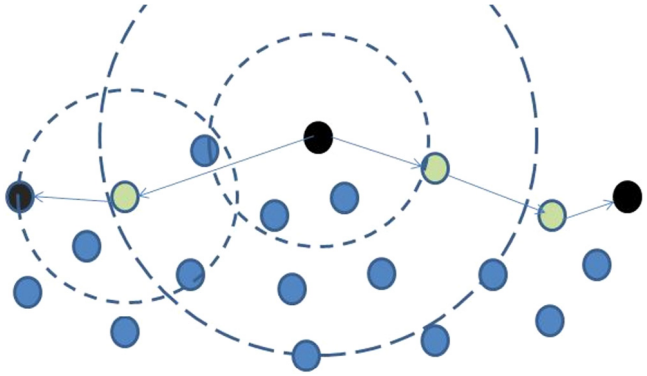

Fig. 1. Identifying connector nodes.

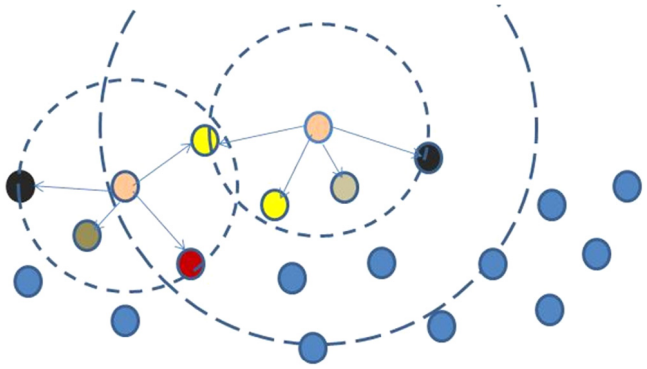

Fig. 2. Packing and coloring in half communication radius balls.

This type of fitting requires running modified shortest path in which the edges are considered between two nodes only if their weight is less than half communication radius. This modification is required to ensure that at least a single connector node is selected in half communication range along the shortest path to acting as the next ball center.

Clearly, a topology having holes will run this algorithm on each boundary with a different leader to obtain packing on all the boundaries. Suppose a local scenario as illustrated in Fig. 1 where a boundary node say bn_i finds out the shortest path to its nearest boundary neighbor and identifies the connector nodes along the path. The part of the closed circuit formed by these nodes looks like bn_{i-1}, cn_{j-1}, cn_j, bn_i, cn_{j+1}, cn_{j+2}, bn_{i+1}. In Fig. 2, bn_i constructs the packing (in one direction) by picking a node cn_{j+1} farthest in its half communication range along bn_{i+1} as the center of the next ball. Similarly, it can be done in both directions by picking one node each along both neighbors bn_{i-1} and bn_{i+1}. The color assignment

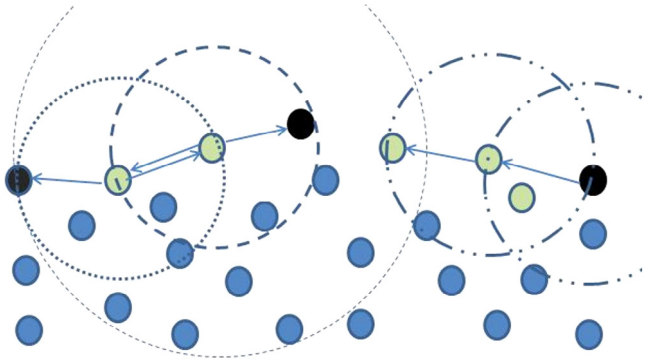

Fig. 3. Simultaneous packing of half communication radius balls along shortest path.

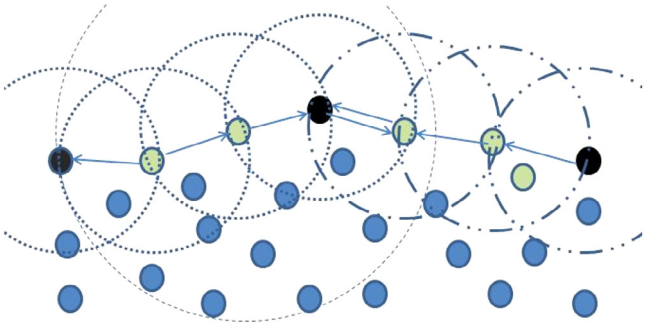

Fig. 4. Increasing packing size.

(random assignment) in half communication radius of both the nodes (after selecting the next ball center) is shown as in Fig. 2.

In the case of color assignment, there can be issues due to nodes falling in the overlapping region. Such node can get two colors to avoid which we impose the condition of being colored only once. Ideally, the minimum of all balls center node's connectivity in half communication range should dictate the number of disjoint connected boundaries, but in practical it is governed by the numbers of successful minimum coloring done by the node with minimum connectivity in half communication range. The efficiency of the packing depends on the density of nodes in the neighborhood. Higher the density of nodes higher the chance of getting a node near the half communication radius distance and lower the overlap (apart from that required for ensuring connectivity) among the fitted balls. These coloring form a domatic partitioning of the boundary region. That is, all the *color i* nodes form a dominating set, further it ensures a connected boundary as for any two *color j* nodes in consecutive balls so that the distance between them is always less than or equal to the communication range *R*. The final number of active boundaries (minimum successful coloring in a neighborhood) could be identified by ball center nodes by running leader election with number of successful coloring as key.

Clearly, for sparse networks, there are chances that this algorithm may fail to provide multiple boundaries as there may not be sufficient number of nodes in the half communication range to get at least a connectivity of 3 in half communication radius balls. How to ensure multiple boundaries in such cases? We will discuss that issue after the following proposed algorithm.

A much elite way to perform domatic partition by using the same shortest path heuristics but by assigning color only along the close circuit formed as follows: (Packing and Coloring Technique 2):

- All the boundary nodes flood the network to know their nearest boundary nodes.
- Each boundary node runs the shortest path algorithm to its nearest neighbors to get two disjoint shortest paths.
- The resulting nodes along the shortest path mark themselves as connectors.
- Each node starts packing by picking a boundary or connectors node farthest in its half communication range. The packing of balls starts with each node having just themselves as the center of the first ball and they select nodes (in both directions) for increasing the size of the packing. A node is added to the packing if its *node id* is smaller than that of the largest *node id* already present in packing. A whole sub-packing is added to a packing if the packing successfully adds a node which is part of the sub-packing. After being added the largest *node id* for the nodes of sub packing is updated.
- A node is added to packing as the center of the ball only if the resulting ball contains no more than two nodes of the same packing. The termination condition thus is the occurring of three nodes in a ball which occurs when packing from both sides meet and there are already two nodes in the last ball causing no further addition.
- After packing is over, perform coloring (can be done by terminating node) by assigning alternate colors to alternate ball center nodes.
- Same color nodes from alternating balls form a boundary.

In Fig. 3, a simultaneous occurring packing scenario is illustrated. Each node selects the farthest node in its half communication radius in both directions to act as the center of next ball. The direction of the arrow shows the direction of packing. In the case of arrows from both nodes, the sub-packing with higher largest *node id* dominates that with lower node id. The next step of packing is shown in Fig. 4 with a sub-packing being dominated by other.

This packing performs coloring in a hasty manner, resulting in a different type of domatic partitioning or in other words different type of disjoint boundary detection. Here, coloring is done after the completion of the fitting of the balls. Each alternate node of a fitting is assigned same color resulting into two coloring or two boundaries. This is a direct implication of Ore's theorem which assures that at least two boundaries exist. Further, more coloring can exist if more disjoint fitting exists. Even if two *color j* nodes are in nonadjacent balls the distance between them ($\frac{R}{2} + \frac{R}{2}$) is always less than or equal to the communication range *R* which ensures the connectivity.

As in *Technique* 2, we find at least two boundaries along the packing. This gives an answer to our previous question i.e., even if the *Technique* 1 fails (i.e., at least one node with no successful coloring in $\frac{R}{2}$), the packing will be already done and we can reassign colors alternately to the nodes to imitate *Technique* 2 and to get two disjoint boundaries. Actually, while *Technique* 2 can work independently and satisfactorily *Technique* 1 should be seen an optimization extension to *Technique* 2. Hence, *boundary* 1 of *Technique* 1 can be interpreted as two disjoint boundaries *boundary* 1*a* and *boundary* 1*b*, while other boundaries are the disjoint substitute for both boundaries 1*a* and 1*b* taken collectively. Algorithm 2 discusses the steps to identify and mark the connector nodes to remove the possible dis-connectivity among boundary nodes. A *in-out* flag is used to denote interior and boundary node. Flag with set $'in'$ denotes the interior nodes and flag with set $'out'$ denotes the boundary nodes.

Algorithm 3 presents the pseudo code of Coloring and Packing *Technique* 1. Similarly, *Technique* 2 can be substituted in algorithm 3 to get different disjoint boundaries.

Algorithm 4 uses the coloring as done by algorithm 3 to identify disjoint boundaries.

Algorithm 2 Identification of Connector nodes

Require: Boundary nodes induced after algorithm 1.
Ensure: Identification of Connector nodes.
if node *in-out* is set *out* **then**
  Flood the network with $< hello, node\_id_{self} >$ to identify other boundary nodes.
  if stored $< hello, node\_id_j >$ **then**
    Reply with $< node\_id_{self} >$ to node with $node\_id_j$.
  end if
end if
if received $< hello, node\_id_j >$ **then**
  if node *in-out* is set *out* **then**
    Reply with $< node\_id_{self} >$.
  else
    if node is still computing *in-out* status **then**
      Store message $< hello, node\_id_j >$.
    end if
  end if
end if
if received $< node\_id_j >$ **then**
  Run distributed Shortest Path (SP) with self as source and node with $node\_id_j$ as destination to obtain shortest path to the node SP_j. Consider edge between nodes only if the distance is less than half of the communication range.
end if
if received $< SP_j >$ **then**
  Sort and store two smallest and disjoint SP_j.
end if

Algorithm 3 Coloring and Packing (using *Technique* 1)

Require: Connected graph formed by Connector nodes and Boundary nodes (algorithm 2.
Ensure: Packing of $\frac{R}{2}$ Balls and Coloring.
Run leader election algorithm in the ring with key $connectivity\_in\_\frac{R}{2}$,$node\_id$ to select leader node L_n with minimum connectivity.
if selected a leader node (L_n) **then**
  Node L_n assign color to self as *color* 0 and sends $< add >$ to a connector node or boundary node (say N_b) farthest in its half communication range.
end if
if received an $< add >$ from Leader **then**
  if already assigned *color i* **then**
    $< deny >$.
  else
    Reply leader with $< accept >$.
  end if
end if
if received an $< accept >$ from a node N_b for an $< add >$ **then**
  Free self as Leader and send $< leader >$ to the N_b. Send $< color\ i >$ to neighbor (say N_i) in half communication radius for $i = 1$ and wait.
  if received $< accept >$ **then**
    Repeat with $< color\ i++ >$ and new neighbor (say N_{i++}).
  else
    if received $< deny >$ **then**
      Repeat with $< color\ i >$ and new neighbor (say N_{i++}).
    end if
  end if
end if
if received an $< color\ i >$ from leader **then**
  if already assigned *color j* **then**
    $< deny >$.
  else
    Reply sender with $< accept >$.
  end if
end if
if received an $< leader >$ from leader **then**
  Mark self as leader.
end if

Algorithm 4 Disjoint boundaries detection

Require: Colored Graph.
Ensure: Disjoint Boundaries(j), where $1 \leq j \geq rad(G)$.
Run leader election algorithm in the ring with key $successful\_coloring\_in_{\frac{R}{2}}$, $node\_id$ to select a leader node with minimum $successful\_coloring\_in_{\frac{R}{2}}$, L_n and to identify minimum number of boundaries. Leader Node L_n flood network with $< no\_of\_boundaries, start\_time, sleep\_interval >$.
if received $< no\_of\_boundaries, start\_time, wake\_interval >$ **then**
  if $color_{self} < no\_of\_boundaries$ **then**
    Estimate sleep time for self taking $start\_time$ as start time for sleep–wake cycle for *color* 0 and considering each boundary remains active for $wake\_interval$. The *color* 0 boundary remain active for $wake\_interval$.
  else
    Set *self* as *colorless* and perform as an interior node.
  end if
end if

5.3. Analysis of disjoint boundary nodes algorithm

In algorithm 1, separator is obtained after applying grid graph (H) of size ($\mathcal{O}\sqrt{N}$), which splits the QUDG graph into $\frac{n}{2}$ size. And, when auxiliary graph (T) is applied on grid graph (H), it splits the graph further into two nonadjacent subgraphs of weight at most $\frac{2}{3}$ of the total weight of the graph (as reported in work [27]). While executing MBFS for boundary detection, we found that enclosing circle is k-hop included graph $G_u^{k\setminus 1} = (V_u^{k\setminus 1}, E_u^{k\setminus 1})$ for any node u. Boundary node detection algorithm (algorithm 1) at worst case requires an asymptotic time complexity of $\mathcal{O}(mn^2)$ where $m = \left|E_u^{k\setminus 1}\right|$, $n = \left|N_u^{k\setminus 1}\right|$ and k is the radius of a graph.

Complexity of algorithm 2 depends on the complexity of shortest path algorithm used and the number of rounds are $\mathcal{O}(BN)$ where BN is the number of boundary nodes detected after algorithm 1. Since, all the boundary nodes can identify their shortest path simultaneously. Complexity of algorithm 3 and algorithm 4 depends on the type of leader election used among the boundary nodes. In addition to the leader election computation, algorithm 3 requires $\mathcal{O}(\delta_{\frac{1}{2}})$ messages per round where $\delta_{\frac{1}{2}}$ is maximum connectivity in half communication range and for algorithm 4, message complexity is $\mathcal{O}(1)$.

For the algorithm 1, the size of every boundary nodes BN in step-6 can be computed with an approximation of $|BN|$=$(3.8opt + 1.2)$ where opt is the size of a minimum dominating set in the (quasi) unit disk graph. For the algorithm 2, the size of connector node can be computed with approximation of $|CN|$=$((1 + \ln 5)opt)$ where opt is the size of a minimum dominating set, from the reported result [18]. Thus, the disjoint dominating sets of boundary region algorithm that have an approximation ratio $((4.8 + \ln 5)opt + 1.2)$ is $|BN| + |CN| = 3.8opt + 1.2 + (1 + \ln 5)opt = (4.8 + \ln 5)opt + 1.2$. Thus, the DBN approach using dominating set algorithm has the approximation ratio $((4.8 + \ln 5)opt + 1.2)$, where opt is the minimum size dominating set [21].

Optimality in terms of Number of Boundary nodes

Consider a case in which boundary nodes are arranged such that the distance between them is just equal to the communication range. In this case, as there will be exactly one node in each communication range, which is just sufficient to maintain the connectivity. We define this as $\mathcal{O}(optimal)$ boundary in terms of number of nodes, i.e., a boundary in which nodes are just connected to communicate throughout the whole boundary.

We have proposed two methods of Packing and Coloring with half communication radius ball $\frac{R}{2}$. For *Technique* 1, we perform $\frac{R}{2}$ balls packing, but when identifying boundary i, one node in

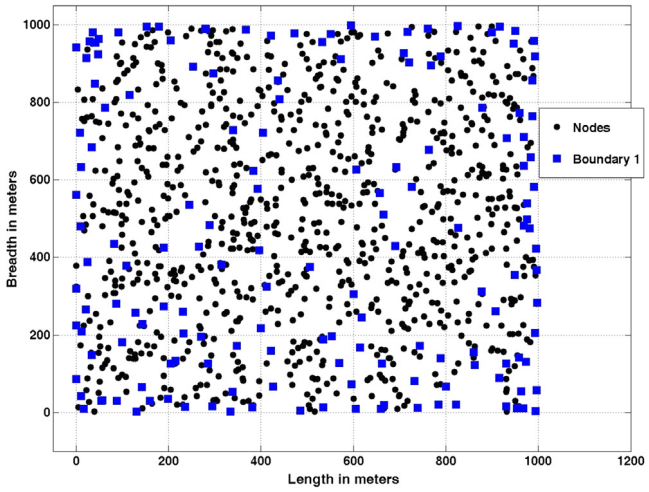

Fig. 5. Boundary at radio range 85 m (using Packing and Coloring *Technique* 1).

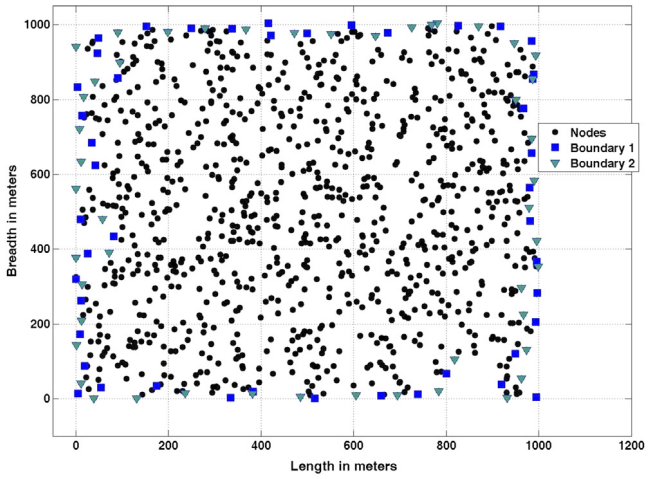

Fig. 6. Boundary at radio range 110 m (using Packing and Coloring *Technique* 1).

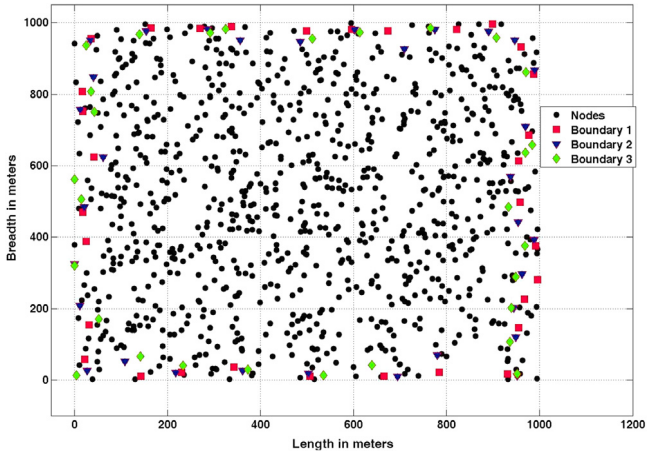

Fig. 7. Node at communication range 150 (using Packing and Coloring *Technique* 1).

alternate ball must act as a boundary *i* node. Hence, a number of boundary nodes present in a communication range is theoretically one. Therefore, using *Technique* 2 our algorithm theoretically can be lower bounded by the number of nodes as $\mathcal{O}(optimal)$. Practically, the number of nodes is quite large due to the cumulative overlap of error for two $\frac{R}{2}$ balls. For *Technique* 2, we perform $\frac{R}{2}$ balls packing, but when identifying boundary *i*, one node in every ball must act as a boundary *i* node. Hence, the number of boundary nodes present in a half communication range is theoretically one. Practically, the balls overlap as there is no guarantee of finding a next ball center at exactly $\frac{R}{2}$ distance. Therefore, using *Technique* 2, our algorithm theoretically can be lower bounded by the number of nodes as $\mathcal{O}(2 * optimal)$. However, as discussed earlier, this is valid only for the *boundary* $i, i \neq 1$ as *boundary* 1 can be considered made up of two disjoint boundaries which can work independently. Hence, for *boundary* 1 even *Technique* 1 is $\mathcal{O}(optimal)$.

Mobility Aspects

Our algorithm can work on sensor networks with disconnected components as each component node can run the algorithm locally. This gives us an advantage over the algorithms [29] where there are some bootstrap beacons which periodically broadcast message for boundary reconsideration. In such cases, *mobility* causing complete disconnection of the graph with components having no beacon nodes may result into no reconfiguration of the boundary while in our algorithm even a totally disconnected component may get its boundary by checking for connectivity periodically by sending a message over the boundary.

6. Simulation result

This section presents the performance and evaluation of our distributed DBN algorithm. We implemented our work in MATLAB. We assumed that node communication is bidirectional in nature with same communication range and nodes are randomly deployed. We evaluated our DBN algorithm for four different scenarios. In the first scenario, communication range is varied over a fixed number of nodes deployed in a fixed area. In the second scenario, we increased the number of nodes at fixed communication range. In the third scenario, we evaluated our DBN algorithm at different typologies and in the final scenario we evaluated the lifetime of boundary nodes.

In next four subsections, we discussed the behavior and trends of boundary node selection for different algorithm and cases. The boundary without packing refers to the boundary obtained after applying algorithm 1. It nearly approximates the other methods of boundary detection available in the literature. Packing without connectors is a reference which tells how many boundary nodes remain after packing randomly the balls taking the output of algorithm 1 as the center of balls (it is a dummy case to identify the number of redundant boundaries while connectivity is given no importance). The second reference for checking our algorithm is the twice optimal number of boundary nodes. It refers to a hypothetical scenario in which a perfect boundary detection algorithm runs on a square topology with nodes arranged perfectly in a grid structure. Another illustrated parameter of our interest is the connectivity of boundary nodes in half communication range. The remaining two plots are for a single boundary (same as *boundary 1a* or *boundary 1b* of DBN using Packing and Coloring *Technique* 1) of DBN using Packing and Coloring *Technique* 2 and a single boundary (*boundary* $i, i \neq 1$) of DBN Using Packing and Coloring *Technique* 1.

6.1. Node communication range

The results in this subsection present the performance of our proposed algorithm at different communication ranges. The number of deployed sensor node is 1000 in an area of 1000 $\times$ 1000 m^2 square. The communication range of nodes is same and varied from 85 m to 200 m with the increment of 5 m, 10 m, 40 m and 50 m. Results in Figs. 5–7 show the boundary nodes detected using Coloring and Packing *Technique* 1 and results in Figs. 8, 9

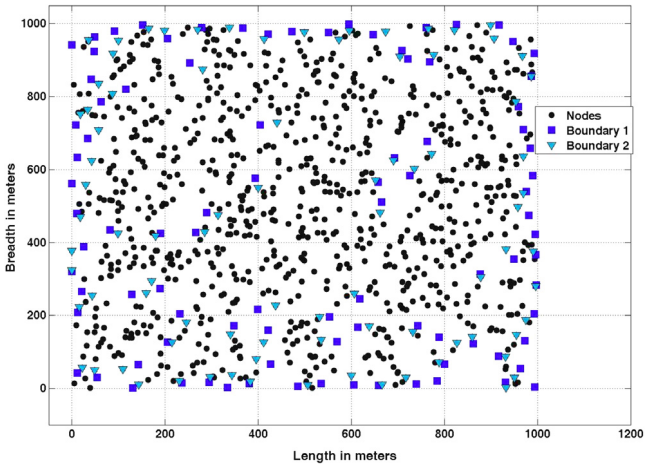

Fig. 8. Node at communication range 85 (using Packing and Coloring *Technique* 2).

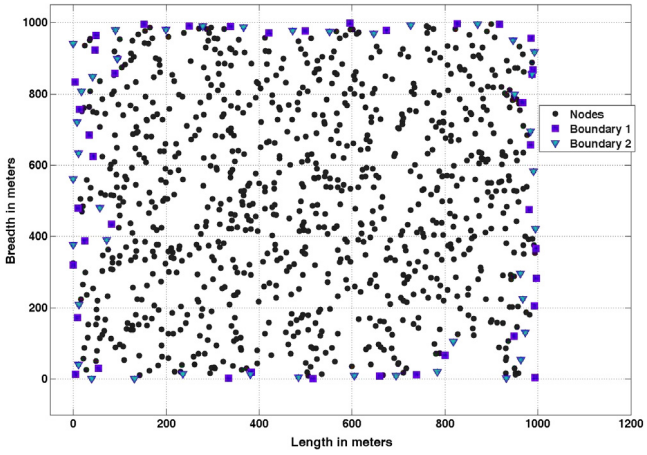

Fig. 9. Node at communication range 110 (using Packing and Coloring *Technique* 2).

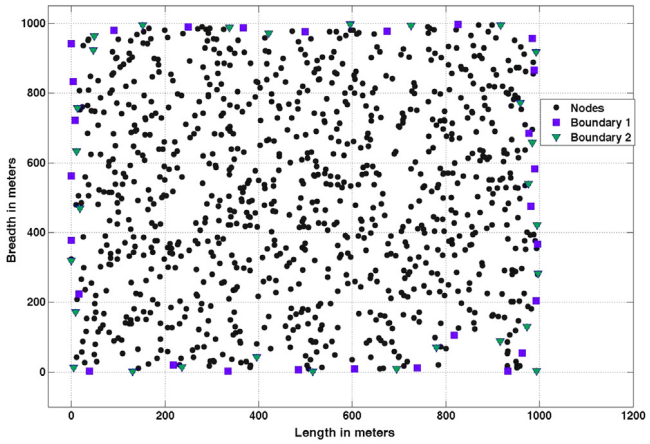

Fig. 10. Node at communication range 150 (using Packing and Coloring *Technique* 2).

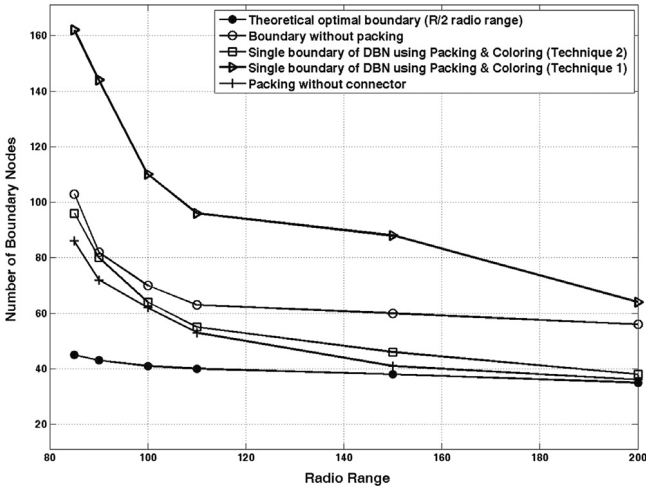

Fig. 11. Optimal boundary detection with respect to communication range.

and 10 show the boundary nodes detected after using Coloring and Packing *Technique* 2. We can observe that there is a decrement in the number of boundary nodes from the Fig. 5 or Fig. 8 or Fig. 7 or Fig. 10. This is because of the removal of interior nodes, which are recognized earlier as hole/network boundaries. Further, the decrements also account for the fact that fewer connector nodes are required as the communication range increases. While a large number of boundary nodes are identified by *Technique* 1. They are as large as twice of that by *Technique* 2 in each boundary. Even then *Technique* 1 is as efficient as *Technique* 2 and even greater because boundary layer 1 of *Technique* 1 is same as boundary layer 1 and layer 2 of *Technique* 2. Hence, it can be reconfigured to achieve as much efficiency as of *Technique* 2 anytime. Here, except the case of communication range being 85 m, in all the remaining cases a higher lifetime than *Technique* 2 can be achieved by *Technique* 1, while in the case it is as equal as *Technique* 2.

The result illustrated in Fig. 11shows the relative performance of the proposed boundary detection algorithm across the communication range in terms of the number of nodes identified. Y axis represents the number of boundary nodes and X axis represents the communication range. The decrease in the number of nodes detected is a common and expected pattern due to increasing the communication range. Clearly, Boundary without Packing has a much higher number of boundary nodes than Packing without Connectors which have lowest but none of them ensures connectivity. Having a higher number of Boundary nodes without Packing needs boundary pruning to remove redundancy at the same time node addition to ensure connectivity. A blind pruning as shown by Packing without Connectors can remove redundancy, but worsen the connectivity problem. A single boundary of DBN using packing and Coloring *Technique* 2 clearly shows the best performance as it ensures the connectivity along with removing redundancy with minimum extra nodes. This algorithm approaches twice optimal packing at high connectivity. When considering a single boundary of DBN using packing and Coloring *Technique* 1, the results seem to be unpleasing as quiet high number of nodes are detected, but as we have discussed how it is always as good as *Technique* 2 or better; further, at higher connectivity (at communication range 200 boundaries detected is 4), the number of boundaries detected is also very high; hence, it decreases the load on each boundary node significantly. Interestingly, we can see that at higher connectivity it approximates the boundary without Packing along with connectivity insurance, which implies that even the boundary i, $i \neq 1$, will have a low number of nodes at high connectivity.

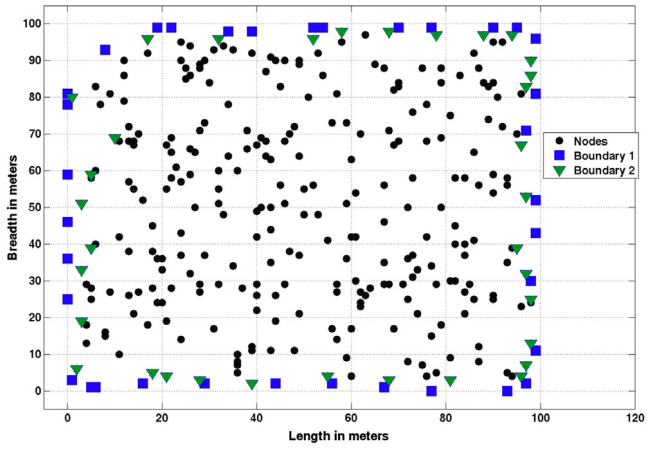

Fig. 12. Boundary detection at 300 node (using Packing and Coloring *Technique* 1).

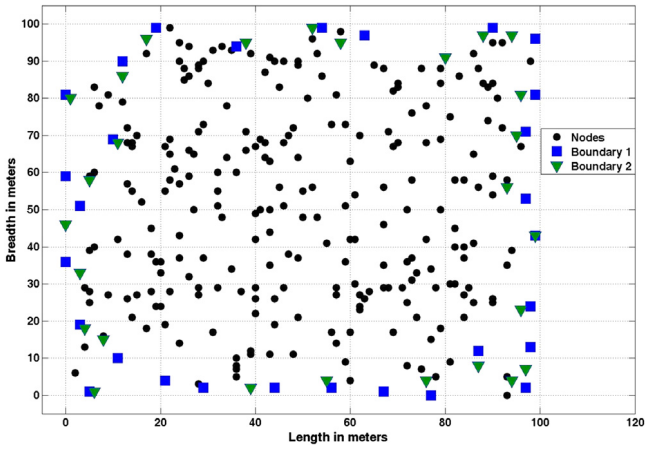

Fig. 14. Boundary detection at 300 node (using Packing and Coloring *Technique* 2).

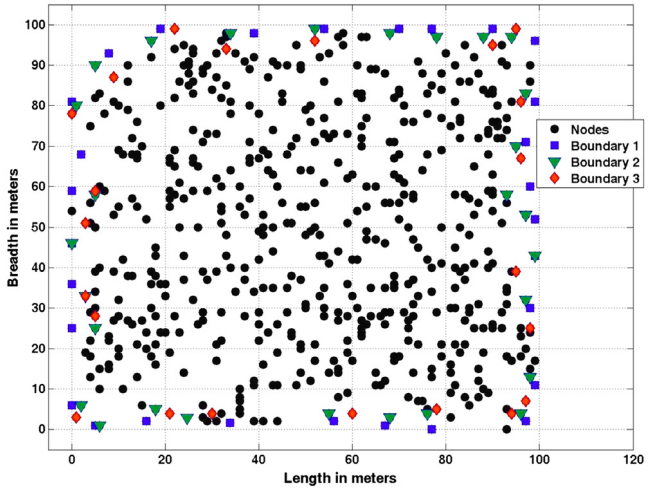

Fig. 13. Boundary detection at 600 node (using Packing and Coloring *Technique* 1).

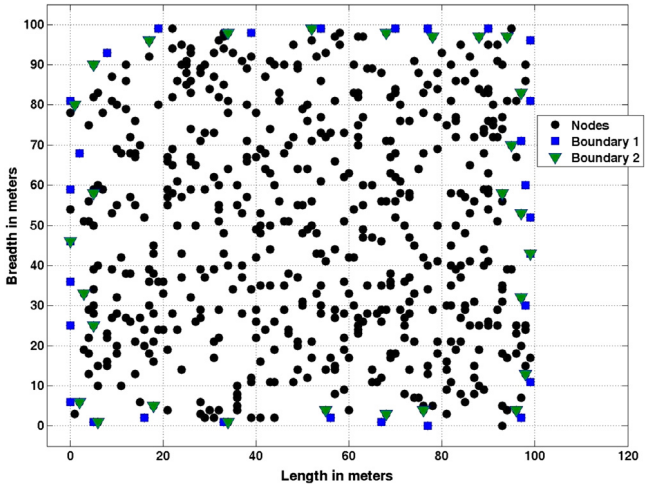

Fig. 15. Boundary detection at 600 node (using Packing and Coloring *Technique* 2).

6.2. Number of nodes

Results, in this section, present the performance of our DBN algorithm at different densities. In this simulation scenario, we placed our nodes in an area of 100 × 100 m^2 while their communication range is made fixed (15 m). Results in Figs. 12 and 13 show the number of boundary nodes and disjoint boundaries when Coloring and Packing *Technique* 1 is used. Similarly, result in Figs. 14 and 15 shows the number of boundary nodes and disjoint boundaries when *Technique* 2 of Coloring and Packing is used.

Based on the definition of twice optimal boundary, the number of nodes detected remains constant with the increase in communication range. The single boundary of DBN using packing and Coloring *Technique* 2 again shows the best performance even in this case where it works as same as a twice optimal boundary detection in terms of the number of nodes. When considering a single boundary of DBN using packing and Coloring *Technique* 1 though the number of boundaries detected is high, but the addition of nodes increases connectivity gradually and at 600 nodes we have as high as 3 boundaries detected. So, at 600 nodes, while nodes of boundary using *Technique* 2 will still have 2 boundaries (say A and B), the *Technique* 1 will not only give boundaries A and B but also two more boundaries with twice the nodes in A or B to share the work. In other words, one can say that *Technique* 1 works like *Technique* 2 except for it distributes the work pressure of nodes of *Technique* 2 to a large number of nodes as the connectivity increases. As evident from the result, our analysis depict that our proposed algorithm requires a minimum connectivity degree of 6 for boundary nodes detection.

6.3. Topology of network

We performed topological testing for our algorithm's durability testing. For the simulation, we created three different topological structure, circle, cross and torus. We deployed 1000 nodes in an area of 100 × 100 m^2 for circle & torus and 150 × 150 m^2 area for cross. We have considered that each node has a communication range of 15 m. The results of our simulation are illustrated in Figs. 16, 18 and 17. The result shows that our proposed DBN (*Technique* 1) algorithm performs well with low false positivity. Result in a torus (Fig. 17) shows that if topology have holes, then they also can be comfortably detected.

6.4. Boundary lifetime

This section shows the impact of disjoint boundaries on the lifetime of boundary nodes. Lifetime is defined as the length of time

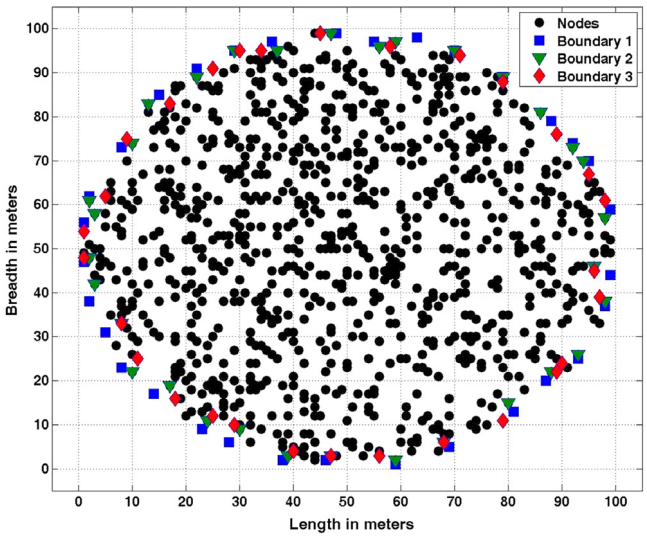

Fig. 16. Boundary detection in circle (using Coloring and Packing *Technique* 1).

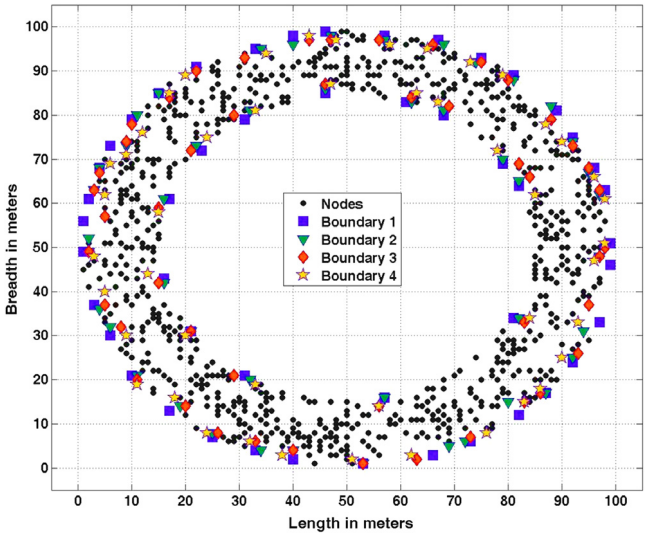

Fig. 17. Boundary detection in torus (using Coloring and Packing *Technique* 1).

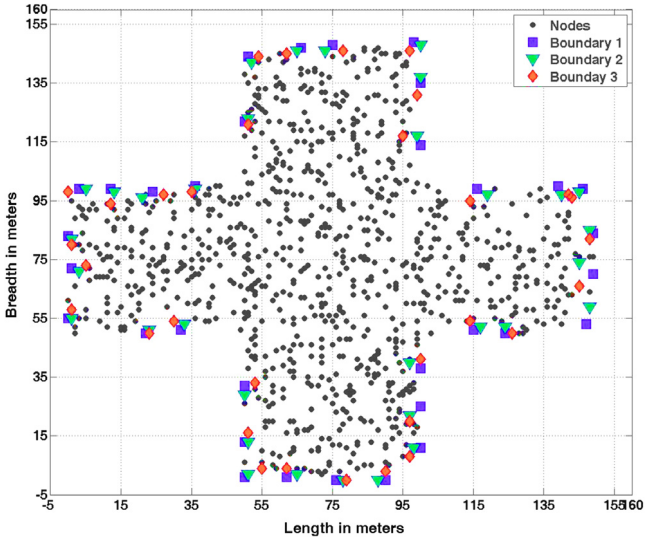

Fig. 18. Boundary detection in cross (using Coloring and Packing *Technique* 1).

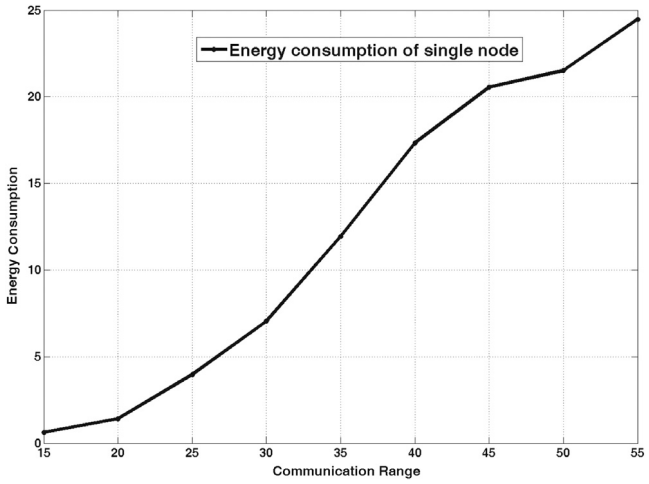

Fig. 19. Energy consumption of single boundary node.

a boundary node is expected to remain alive [15]. The proposed DBN algorithm detects multiple (i.e., disjoint sets) boundaries. Hence, for every boundary, there is the other boundary in its neighborhood, which can work as a boundary node while it sleeps for some duration. During sleep mode of boundary nodes, surveillance duty is shifted to other disjoint boundary nodes. Therefore, the duty cycle of the boundary nodes is decreased by 50% theoretically in the case of 2 boundaries, 33% in case of three and so on. To analyze the energy consumption in *WSN* scenario, we perform a simulation which consists of 600 nodes in an area of 100×100 m^2 (as shown in Fig. 13).

The simulation scenario assumes that the communication range follows d^2 model range of energy consumption and is almost constant in the neighborhood to emphasize on the algorithmic energy consumption rather than the physical conditions. *Y* axis in Fig. 19 shows the energy consumption and *X* axis represents the communication range of the node. Result in Fig. 19, also shows that with the increase in communication range of node increases the energy consumption of node.

None of the reported work on boundary detection focuses on boundary node lifetime maximization. Hence, we compare our algorithm with proposed Coloring and Packing *Technique* 1 (where the total number of identified disjoint boundaries is 2) algorithm and without Coloring and Packing technique algorithm. The result obtained after the simulation is illustrated in Fig. 20. The result shows that algorithm with Coloring and Packing *Technique* 1 consumes 35.71% of less energy with respect to without coloring and packing technique algorithms.

Proposed DBN algorithm executes at the start-up of the network setup. However, the unbalanced energy depletion among all the nodes in the network is unavoidable. The scenario may occur like excessive processing (routing process, continues sensing, technical glitch) nodes may run out of energy. Such scenario creates a local holes problem [28,32]. When surveillance and Greedy routing (dead end [23–25]) is performed, energy consumption becomes a serious issue around the hole boundary. So, to overcome energy depletion problem due to local holes creation, DBN algorithm can be executed periodically. The proposed DBN algorithm detects multiple (i.e., disjoint sets) boundaries. Hence, for every boundary, there is another boundary in its neighbor, which can share duty and reduce the energy consumption of hole boundary.

7. Experimental setup and results

For experimental analysis of our DBN algorithm (identification of network boundary and interior nodes), we have used TelosB

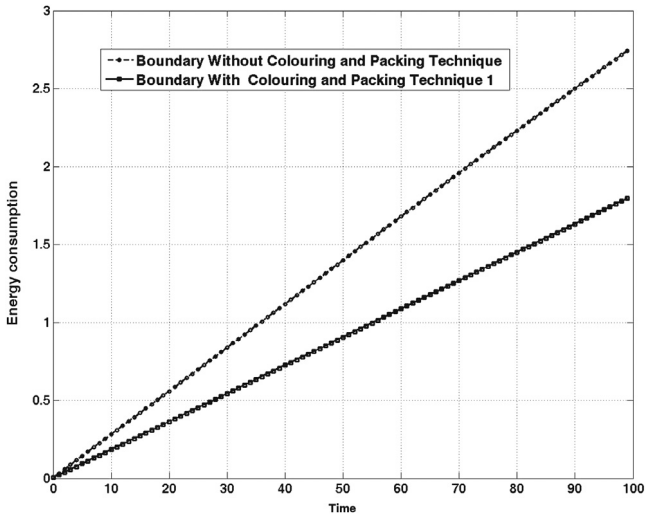

Fig. 20. Boundary lifetime comparison between DBN and without DBN.

```
typedef nx_struct RssiMsg {
 nx_int16_t node_id;   //node_id = ID of sending node (2 byte)
 nx_int16_t snode_id;  //snode_id = ID of second node (2 byte)
 nx_int16_t dis;       //dis = distance between sending node and neighbor node (2byte)
} RssiMsg;
```

Fig. 21. Message structure for distance sharing and boundary detection.

motes based testbed. The testbed setup is build using 8 TelosB motes, and the algorithm was implemented using nesC programming running on TinyOS operating system. Each TelosB nodes were assigned with unique-id from 1 to 8. The message size was kept as 6 bytes, and the message structure is defined as shown in Fig. 21:

Node id shows the sender node id, which is stored in variable *TOS_NODE_ID*. *Snodeid* field shows the neighbor node id. Distance field shows the hops between sender and other node.

We have used four timers in our program. The first timer is used for sending messages, the second timer is used for ranging i.e. distance detection between nodes, the third timer is used for boundary node detection and the fourth timer is used for packing algorithm. We assume that all the nodes are in communication range and transmitting/receiving messages at an interval of 1 s.

Figs. 21 and 22 illustrate the message structure used in disjoint boundary selections. All the nodes send message $<$ *Nodeid*, *null*, *null* $>$ to its neighbors at the beginning of the experiment.

Experimental Results

Each individual node executes the event *Timer*0 for detecting itself as a boundary or interior in distributed manner. The timer of 10 s is set. We assume that nodes have some mobility we need to refresh ranging information using the timer. Event *Timer*0 is used to test whether the node is interior or not, according to our algorithm (MBFS). Function *setLeds* is used for switching *leds*, as it helps in discrimination between the boundary and interior motes. For convenience, boundary nodes are marked as red while interior nodes are given the blue color. Fig. 23 illustrates the snapshot of our conducted experiment. After some time, *SendTimer* is stopped and thus there will be no further calculation for hops. Then, boundary nodes will set the boolean variable *BOUNDARY* as 1 and interior nodes will set it to 0.

Once nodes are declared as boundary nodes and interior node, our algorithm identifies the redundant nodes on boundaries to

```
typedef nx_struct PckMsg {
nx_int16_t node_id;  // node_id = ID of sending node of 2 byte
nx_int8_t color      // color is Boolean variable, if set to 1 then yellow else red.
} PckMsg;
```

Fig. 22. Message structure used in disjoint boundary selection.

Fig. 23. Snapshot of our experiment.

find out the minimum of a boundary node set needed for network boundary. All the experiment is performed in an indoor environment. We have tested and verified our algorithm on different topologies and rectangular grid and found detect network boundary successfully. However, the algorithm is converging quicker in certain topologies compared to grid topology. We performed two different sets of experiments for analyzing success rate of our algorithm. We have calibrated the experiment for indoor with the nodes placed beyond 24–30 m apart.

The first experiment is performed by placing TelosB motes in a grid topology in which one node is placed in the middle and another seven nodes were placed around enclosing that node at a distance of 10 m. Maximum radio range of motes is observed as 24 m. We observe that our algorithm successfully detected all the boundary nodes and interior nodes in 90 s. However, this convergence time is reduced to 20 s on experimentation with the random topology. To test our algorithm's accuracy and convergence time, we place nodes in square topology, in such a way that two squares are formed and bigger square covers the smaller square. Bigger square is formed in an area of 18 $\times$ 18 m and smaller square is formed in 10 $\times$10 m. 4 nodes are used for outer square placed at the corners and 4 nodes are used as interior node placed at the corners of the smaller square of 10 $\times$ 10 m. We have used these scenarios to test our algorithm's operability with both interior node and boundary nodes, and the result obtained is shown in Table 1.

8. Conclusion and future work

In this paper, we have proposed a novel solution for the challenging problem of boundary region detection and maximization of boundary node lifetime. We have used the QUDG connectivity model for boundary region detection and partition the detected boundary region of the network into disjoint dominating sets. By rotating the disjoint dominating sets of boundary region in the network, the duty cycle of the boundary nodes decreases, which results to maximize the lifetimes of boundary nodes. Simulation results show that our proposed disjoint boundary node (DBN) algorithm shows that the boundary node saves 35.71% of energy lesser as compared to single layer boundary node. Our approach is implemented and tested on telosB motes. Since the impact of link

Table 1
Convergence rate of algorithm.

Time (s)	% of accurate boundary detected	% of accurate interior node detected	% convergence
15	50	25	37.5
30	50	50	50
45	50	50	50
65	50	75	62.5
82	75	75	75
110	75	100	87.25
145	100	100	100

and node failure network is not analyzed rigorously in this work, we have included that the future scope of work is to study our proposed approach in the scenario where node and link failures occur frequently.

References

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci, A survey on sensor networks, IEEE Commun. Mag. 40 (8) (2002) 102–114.

[2] Antonio Caruso, Stefano Chessa, Swades De, Alessandro Urpi, GPS free coordinate assignment and routing in wireless sensor networks, in: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, in: Proceedings IEEE, vol. 1, IEEE, 2005, pp. 150–160.

[3] Jianer Chen, Anxiao Andrew Jiang, Iyad A. Kanj, Ge Xia, Fenghui Zhang, Separability and topology control of quasi unit disk graphs, Wirel. Netw. 17 (1) (2011) 53–67.

[4] Jitender S. Deogun, Saket Das, Haitham S. Hamza, Steve Goddard, An algorithm for boundary discovery in wireless sensor networks, in: International Conference on High-Performance Computing, Springer Berlin Heidelberg, 2005, pp. 343–352.

[5] Ahmed Faheem, Reino Virrankoski, Mohammed Elmusrati, Improving RSSI based distance estimation for 802.15. 4 wireless sensor networks, in: Wireless Information Technology and Systems, ICWITS, 2010 IEEE International Conference on, IEEE, 2010, pp. 1–4.

[6] Qing Fang, Jie Gao, Leonidas J. Guibas, Locating and bypassing routing holes in sensor networks, in: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. 4, IEEE, 2004.

[7] Qing Fang, Jie Gao, Leonidas J. Guibas, Locating and bypassing holes in sensor networks, Mob. Netw. Appl. 11 (2) (2006) 187–200.

[8] Jay Farrell, Aided Navigation: GPS with High Rate Sensors, McGraw-Hill, Inc., 2008.

[9] Marwan Fayed, Hussein T. Mouftah, Localised convex hulls to identify boundary nodes in sensor networks, Int. J. Sens. Netw. 5 (2) (2009) 112–125.

[10] Sándor P. Fekete, Michael Kaufmann, Alexander Kröller, Katharina Lehmann, A new approach for boundary recognition in geometric sensor networks 2005. arXiv preprint cs/0508006.

[11] Sándor P. Fekete, Alexander Kröller, Dennis Pfisterer, Stefan Fischer, Carsten Buschmann, Neighborhood-based topology recognition in sensor networks, in: International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, Springer Berlin Heidelberg, 2004, pp. 123–136.

[12] Stefan Funke, Topological hole detection in wireless sensor networks and its applications, in: Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing, ACM, 2005, pp. 44–53.

[13] Stefan Funke, Christian Klein, Hole detection or: how much geometry hides in connectivity? in: Proceedings of the Twenty-Second Annual Symposium on Computational Geometry, ACM, 2006, pp. 377–385.

[14] Robert Ghrist, Abubakr Muhammad, Coverage and hole-detection in sensor networks via homology, in: Information Processing in Sensor Networks, 2005 IPSN 2005 Fourth International Symposium on, IEEE, 2005, pp. 254–260.

[15] Deokwoo Jung, Thiago Teixeira, Andreas Savvides, Sensor node lifetime analysis: Models and tools, ACM Trans. Sens. Netw. 5 (1) (2009) 3.

[16] I. Khan, H. Mokhtar, M. Merabti, A survey of boundary detection algorithms for sensor networks, in: Proceedings of the 9th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, 2008.

[17] Alexander Kröller, Sándor P. Fekete, Dennis Pfisterer, Stefan Fischer, Deterministic boundary recognition and topology extraction for large sensor networks, in: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, Society for Industrial and Applied Mathematics, 2006, pp. 1000–1009.

[18] Yingshu Li, My T. Thai, Feng Wang, ChihWei Yi, PengJun Wan, DingZhu Du, On greedy construction of connected dominating sets in wireless networks, Wirel. Commun. Mob. Comput. 5 (8) (2005) 927–932.

[19] Richard J. Lipton, Robert Endre Tarjan, A separator theorem for planar graphs, SIAM J. Appl. Math. 36 (2) (1979) 177–189.

[20] Abdelhamid Mammeri, Azzedine Boukerche, Zongzhi Tang, A real-time lane marking localization, racking and communication system, Comput. Commun. 73 (2016) 132–143.

[21] Rajiv Misra, Chittaranjan Mandal, Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks, IEEE Trans. Parallel Distrib. Syst. 21 (3) (2010) 292–302.

[22] Ahmed Mostefaoui, Azzedine Boukerche, M.A. Merzoug, Mahmoud Melkemi, A scalable approach for serial data fusion in wireless sensor networks, Comput. Netw. 79 (2015) 103–119.

[23] Ahmed Mostefaoui, Mahmoud Melkemi, Azzedine Boukerche, Routing through holes in wireless sensor networks, in: Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM, 2012, pp. 395–402.

[24] Ahmed Mostefaoui, Mahmoud Melkemi, Azzedine Boukerche, Localized routing approach to bypass holes in wireless sensor networks, EEE Trans. Comput. 63 (12) (2014) 3053–3065.

[25] Horacio A.B.F. de Oliveira, Azzedine Boukerche, Daniel L. Guidoni, Eduardo F. Nakamura, Raquel A.F. Mini, Antonio A.F. Loureiro, An enhanced location-free greedy forward algorithm with hole bypass capability in wireless sensor networks, J. Parallel Distrib. Comput. 77 (2015) 1–10.

[26] Sriram V. Pemmaraju, Imran A. Pirwani, Energy conservation via domatic partitions, in: Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2006, pp. 143–154.

[27] Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, Seth Teller, Anchor-free distributed localization in sensor networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, ACM, 2003, pp. 340–341.

[28] Heitor S. Ramos, Azzedine Boukerche, Alyson L.C. Oliveira, Alejandro C. Frery, Eduardo M.R. Oliveira, Antonio A.F. Loureiro, On the deployment of large-scale wireless sensor networks considering the energy hole problem, Comput. Netw. 110 (2016) 154–167.

[29] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, Ion Stoica, Geographic routing without location information, in: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, ACM, 2003, pp. 96–108.

[30] Prasan Kumar Sahoo, Kun-Ying Hsieh, Jang-Ping Sheu, Boundary node selection and target detection in wireless sensor network, in: Wireless and Optical Communications Networks, 2007 WOCN'07. IFIP International Conference on, IEEE, 2007, pp. 1–5.

[31] Yue Wang, Jie Gao, Joseph S.B. Mitchell, Boundary recognition in sensor networks by topological methods, in: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, ACM, 2006, pp. 122–133.

[32] Xiaobing Wu, Guihai Chen, Sajal K. Das, Avoiding energy holes in wireless sensor networks with nonuniform node distribution, IEEE Trans. Parallel Distrib. Syst. 19 (5) (2008) 710–720.

[33] Hongyu Zhou, Hongyi Wu, Miao Jin, A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 1602–1610.

Shailendra Shukla received the M.S. degree in Information Security from Indian Institute of Information Technology in 2009 and completed his Ph.D. in the area of wireless sensor networks from Indian Institute of Technology Patna in 2015. Currently, he is working as assistant professor in computer science department at the Jaypee University of Information Technology. His research interest includes Advanced Computer Networks, Wireless Networks, Security and Internet of Things.

Rajiv Misra received the BE degree in computer science from the Motilal Nehru National Institute of Technology (MNIT), Allahabad, in 1987 and the M-Tech degree in computer science and engineering from the Indian Institute of Technology (IIT), Bombay, in 1989. He completed his Ph.D. work in the area of mobile computing at IIT Kharagpur. He is currently an assistant professor in the Computer Science and Engineering Department, IIT Patna. His research interests include algorithms for ad hoc and sensor networks.

Animesh Prasad received his B-Tech degree in computer science from the Indian Institute of Technology Patna in 2014 and currently, he is Ph.D. candidate at School of Computing, National University of Singapore.