


# Multiple token-based neighbor discovery for directional sensor networks

Shamanth Nagaraju<sup>1</sup>  | Lucy J. Gudino<sup>2</sup> | Nipun Sood<sup>1</sup> | Jasmine G. Chandran<sup>1</sup> | V. Sreejith<sup>1</sup>

<sup>1</sup>Department of Computer Science and Information Systems, BITS Pilani, K. K. Birla Goa Campus, Goa, India

<sup>2</sup>Department of Computer Science and Information Systems, BITS Pilani, Pilani, Rajasthan, India

## Correspondence

Shamanth Nagaraju, Department of Computer Science and Information Systems, BITS Pilani, K. K. Birla Goa Campus, Goa, India.  
Email: shamanth.nagaraj@gmail.com

Directional sensor networks (DSNs) can significantly improve the performance of a network by employing energy efficient communication protocols. Neighbor discovery is a vital part of medium access control (MAC) and routing protocol, which influences the establishment of communication between neighboring nodes. Neighbor discovery is a challenging task in DSNs due to the limited coverage provided by directional antennas. Furthermore, in these networks, communication can only take place when the beams of the directional antennas are pointed toward each other. In this article, we propose a novel multiple token-based neighbor discovery (MuND) protocol, in which multiple tokens are transmitted based on an area exploration algorithm. The performance of the protocol is evaluated using the Cooja simulator. The simulation results reveal that the proposed MuND protocol achieves lower neighbor discovery latency, with a 100% neighbor discovery ratio, and has a relatively low communication overhead and low energy consumption.

## KEYWORDS

directional sensor networks, multiple token, neighbor discovery, sector antenna

## 1 | INTRODUCTION

In the field of wireless sensor networks (WSNs), directional sensor networks (DSNs) provide a great scope for performance improvements. The perks of using wireless networks with directional antennas include lower interference, diminished power requirements, longer transmission range, better spatial reuse and so on [1,2]. DSNs employ directional antennas at the receiver and the transmitter, where communications transpire only when the two associated beams intersect. DSNs have the advantages of better signal quality, better energy efficiency, better routing performance, and higher capacity [3]. In a traditional network employing omnidirectional antennas, establishing communication between neighboring nodes is trivial. It can be accomplished simply via the

broadcast mechanism. However, establishing communication between neighboring nodes in DSNs is more challenging due to the limited coverage provided by the directional antennas, giving rise to the problem of neighbor discovery. Neighbor discovery poses the difficult challenge of determining when and where to point the antenna beams to achieve data transmission and reception between neighbors [4].

Neighbor discovery protocols are mainly categorized into two types: informed discovery and blind discovery [5]. The former involves localization of the neighboring nodes and the latter involves the transmission of probes and listening for the same in random directions [6,7]. Informed discovery retrieves information about the relative positions of neighbors and stores it using angle of arrival (AoA) caching [8], location table maintenance [9], gossip-based algorithm [10],

and request-to-send (RTS) multi-hop routing [11]. The information stored is revised whenever the state of any neighbor is detected to have changed. This incurs a very large overhead, making the technique inefficient [5]. The other type of neighbor discovery, blind discovery, is a more complex procedure, where information about neighbors can only be obtained during a limited period.

In addition to finding an adequate and competent algorithm for neighbor discovery, it is also essential for the algorithm to address issues like deafness, where signal packets are lost if the receiver node's antenna is not oriented toward the sender node's antenna [12]. Deafness can cause collision of packets, leading to increased energy consumption, poor quality of service (QoS), poor packet delivery ratios and so on [13]. An efficient neighbor discovery algorithm can mitigate the effects of deafness by sending packets in a particular sector for a predetermined period of time so that all the intended receivers can be notified beforehand and discovered.

The rest of the paper is organized as follows. An overview of related works is presented in Section 2. A novel multiple token-based neighbor discovery (MuND) protocol is proposed in Section 3. Section 4 presents an evaluation of the performance of the proposed MuND protocol in comparison with that of well-known protocols and a conclusion drawn in Section 5.

## 2 | RELATED WORK

The existing literature on neighbor discovery protocols can be broadly categorized into three classes: one-way [14–16], two-way [17–19], and three-way handshaking [20] mechanisms. In one-way broadcast mechanism, nodes broadcast their locations periodically. Upon receiving at least one such packet, a neighboring node becomes aware of the presence of sender node. In one-way neighbor discovery, the sender node is not able to gauge the correct timing to steer its beam toward the neighboring node for further communication after discovery. This condition engenders the need for a feedback mechanism, which can be accomplished via the two-way or the three-way handshaking mechanism [21].

In two-way handshaking neighbor discovery mechanism, upon receiving a *HELLO* packet from the sender node, the receiver node acknowledges the sender node with a *REPLY* packet, thus facilitating the discovery of both nodes. A scan-based asynchronous neighbor discovery (SBAN) protocol has been proposed in [17]. A slow scan-based *HELLO* mechanism and a fast *REPLY* mechanism are introduced in this scheme to speed up the handshaking process. However, it does not address the issue of collisions between two or more *REPLY* packets that have been transmitted to the same sender node antenna. This issue of collision can be addressed via randomized two-way neighbor discovery [18], which

employs selective feedback. However, both SBAN and randomized two-way neighbor discovery protocols suffer from the lack of beam synchronization. As a result, the reception of *HELLO* or *REPLY* packets is not guaranteed. Beam synchronization is achieved in the collaborative neighbor discovery (COND) protocol [19,22]. Direct and indirect COND protocols propose a distributed and collaborative method of neighbor discovery which is meant to drastically reduce neighbor discovery latency. In direct COND, each node polls to directly discover its neighbors and in indirect COND, each node updates its neighbor table by using neighbor information obtained from other nodes. COND protocols are assumption-based as they require assumptions regarding the density of nodes in the deployed region. They are computationally quite complex.

In order to overcome the aforementioned drawbacks of the two-way handshaking mechanism, such as the lack of beam synchronization and collision avoidance, a three-way handshaking mechanism can be employed. Sectorized antenna neighbor discovery (SAND) [20] is a well-known three-way handshaking neighbor discovery protocol. SAND follows a centralized method, where neighboring nodes are discovered in a serialized manner within a predetermined period of time. In this approach,  $K$ -sectorized antennas are used both at the receiver and at the transmitter, which aids the process of neighbor discovery. Serialization is implemented with the help of a token, which is held by a central node. The centralization of the entire process helps in managing and synchronizing the nodes. The process involves passing a token among the nodes, where only the token holder can perform neighbor discovery, through a *HELLO-REPLY* mechanism. Each node maintains a neighbor table and relays this information to the central node. The major drawback of this approach is that, due to the serialization approach, the neighbor discovery latency increases proportionally with network size.

To overcome the abovementioned drawbacks of the existing neighbor discovery protocols, we propose a three-way handshaking neighbor discovery protocol which employs multiple tokens to reduce discovery latency and to solve the deafness problem.

## 3 | PROPOSED APPROACH

While numerous versions and variations of neighbor discovery protocols in DSNs have been published, major challenges still persist. While protocols like SAND employ a centralized method of 100% neighbor discovery, the serialized mechanism increases the time required for complete discovery. Direct and indirect COND protocols provide a fully distributed method of neighbor discovery, achieving around 90% neighbor discovery in a relatively short period of time [19], but does not guarantee complete neighbor discovery.

The proposed protocol MuND adopts a three-way handshaking mechanism and uses multiple tokens instead of a single one as in case of SAND. The use of multiple tokens helps in reducing the total time taken to complete the neighbor discovery process without compromising on the neighbor discovery ratio. Initially, the sink node holds “T” tokens and initiates a neighbor discovery process to discover nodes present around it. From the sink node, “T” tokens are relayed along “T” different paths obtained via the token path planning (TPP) algorithm explained in Algorithm 1. For example, the path obtained via the TPP algorithm is denoted by the sequence  $\text{sink\_node} \rightarrow \text{farthest\_destination\_cell1} \rightarrow \text{farthest\_destination\_cell2} \dots \rightarrow \text{sink\_node}$ , where  $\text{farthest\_destination\_cell1}$  represents the cell that is the farthest from sink node,  $\text{farthest\_destination\_cell2}$  is cell that is the farthest from  $\text{farthest\_destination\_cell1}$  etc. The traversal path of each token from one destination cell to another destination cell may contain several nodes and best candidate node is chosen as the token holder node. Then, every token holder node, in turn, performs neighbor discovery. Before relaying the token toward the designated  $\text{farthest\_destination}$  cell, an appropriate sector in the sectorized antenna at the token holder node is chosen for sending the token to one of the candidate nodes residing in that particular sector. Once a token reaches the  $\text{farthest\_destination}$  cell, the subsequent  $\text{farthest\_destination}$  cell coordinates in the given path are used for subsequent traversal. This process is continued until all  $\text{farthest\_destination}$  cells have been traversed by individual tokens. When the entire area has been traversed, the tokens return to the sink node and the neighbor discovery process is completed.

During this process, non-token holder nodes operate in *Fast-Scan* mode, where they switch between their available sectors at a relatively faster rate. The token holder nodes operate in *Slow-Scan* mode and transmit *HELLO* packets, switching from one sector to another only when the non-token holder nodes complete one complete cycle of switching over all available  $K$ -sectors than is the non-token holder nodes complete one round of discovery. This mechanism ensures that each non-token holder node's sector/beam intersects the token holder node's sector/beam at a certain point of time. In *Fast-Scan* mode, the nodes respond to *HELLO* packets with *REPLY* packets. Figure 1 shows the time line of the proposed MuND protocol. As depicted in Figure 1, the MuND protocol operates in six phases:

1. Token path planning
2. Fetching the destination coordinates
3. Neighbor discovery
4. Bounded region calculation
5. Best candidate node selection for token passing
6. Token passing

### 3.1 | Token path planning

In the MuND protocol, tokens are not relayed arbitrarily among nodes. Instead, every token follows a predefined path. The path that a particular token follows is determined using the TPP algorithm which is based on the Max-gain area exploration algorithm [23]. The traversal path for the token determined using the TPP algorithm helps to carry out neighbor discovery efficiently. Algorithm 1 describes the TPP algorithm and it is also illustrated in Figure 2 by considering the example of a system with four tokens. The parameters used in the TPP algorithm are presented in Table 1.

We assume that sensor nodes are deployed in a polygonal area DR which fits in the rectangular region  $R$ .  $R$  is split into cells  $C_{m,n}$ , as depicted in Figure 2, where the pair  $(m, n)$  denotes the row and column position of a cell in  $R$ . Each cell  $C_{m,n}$  occupies unit area  $a_C$  and is associated with the color value  $CV(C_{m,n})$  and the utility value  $U(C_{m,n})$  with respect to a particular token. The  $CV(C_{m,n})$  can be either WHITE, BLACK, or GREY, indicating that the cell has been explored, is unexplored, or is in the path between a token's source and destination cells, respectively. The utility value  $U(C_{m,n})$ , designated to each cell, aids in averting collision among tokens as well as avoiding redundant traversals. To audit the cell's status, the sink node maintains the color value and utility value of each cell in a common map,  $M$ . By using the information available in the map,  $M$ , the TPP algorithm selects the path of traversal for each token. The token's traversal path may contain several  $\text{farthest\_destination}$  cells. For instance,  $t1$ ,  $t2$ ,  $t3$ , and  $t4$  are four tokens used in the algorithm as depicted in Figure 2. The path for these four tokens is as follows:  $t1: \text{sink} \rightarrow \dots C_{1,1}$ ,  $t2: \text{sink} \rightarrow \dots C_{1,16}$ ,  $t3: \text{sink} \rightarrow \dots C_{16,1}$ , and  $t4: \text{sink} \rightarrow \dots C_{16,16}$ , where  $\text{sink}$  indicates the sink node located at  $C_{9,9}$  from which the tokens are deployed and  $C_{i,j}$  represents the cell with coordinates  $(i, j)$ .

A cell's utility value,  $U(C_{m,n})$ , depends on the density of BLACK cells  $n_b$  and distance utility value  $DU(C_{m,n})$  between the position of the current cell position and that of the  $\text{farthest\_destination}$  cell. The utility value  $U(C_{m,n})$ , assigned to a cell with respect to token  $t_k$ , is computed as follows.

$$U(C_{m,n})_k = \alpha \cdot (n_b)_k - \left( \left( \sum_{i=1}^T (DU_{m,n})_i \right) - (DU_{m,n})_k \right) \quad (1)$$

where  $k$  varies from 1 to  $T$ ,  $T$  is the total number of tokens and  $\alpha$  is a constant.  $DU(C_{m,n})$ , assigned to each cell with respect to token  $t_k$ , is given by

$$DU(C_{m,n})_k = \begin{cases} \left( 1 - \frac{d(CA_k, C_{m,n})}{R_k} \right) & \text{if } d(CA_k, C_{m,n}) < R_k, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

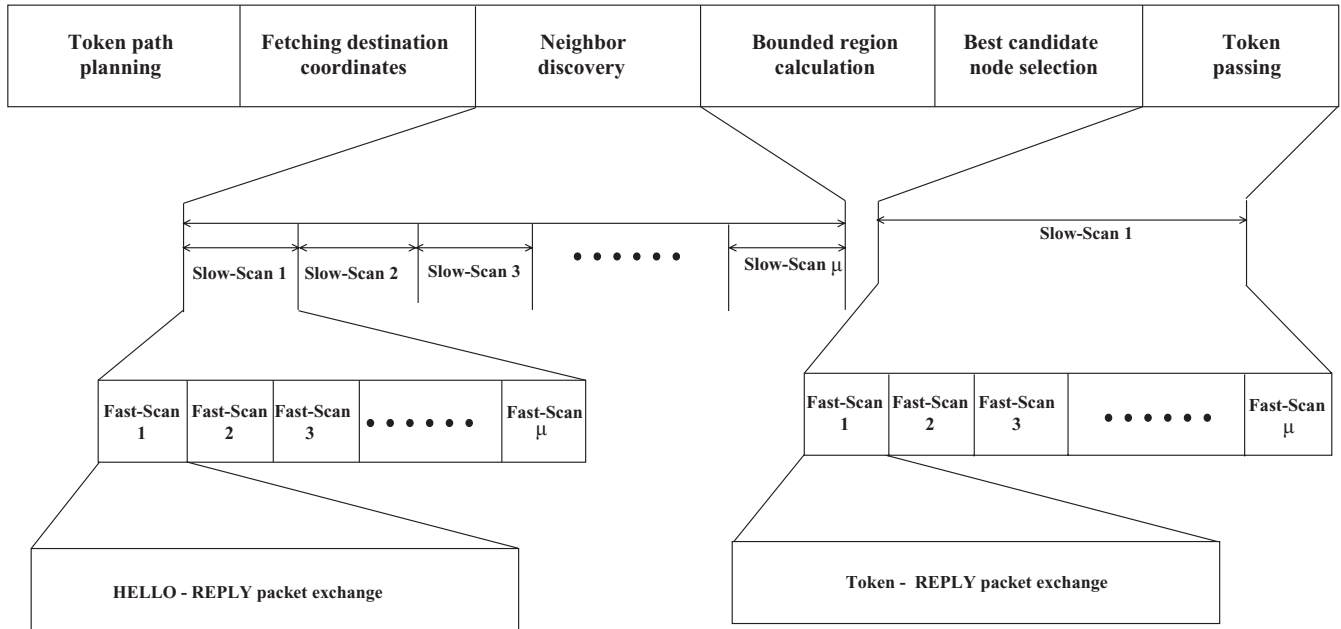


FIGURE 1 Time line of multiple token-based neighbor discovery protocol

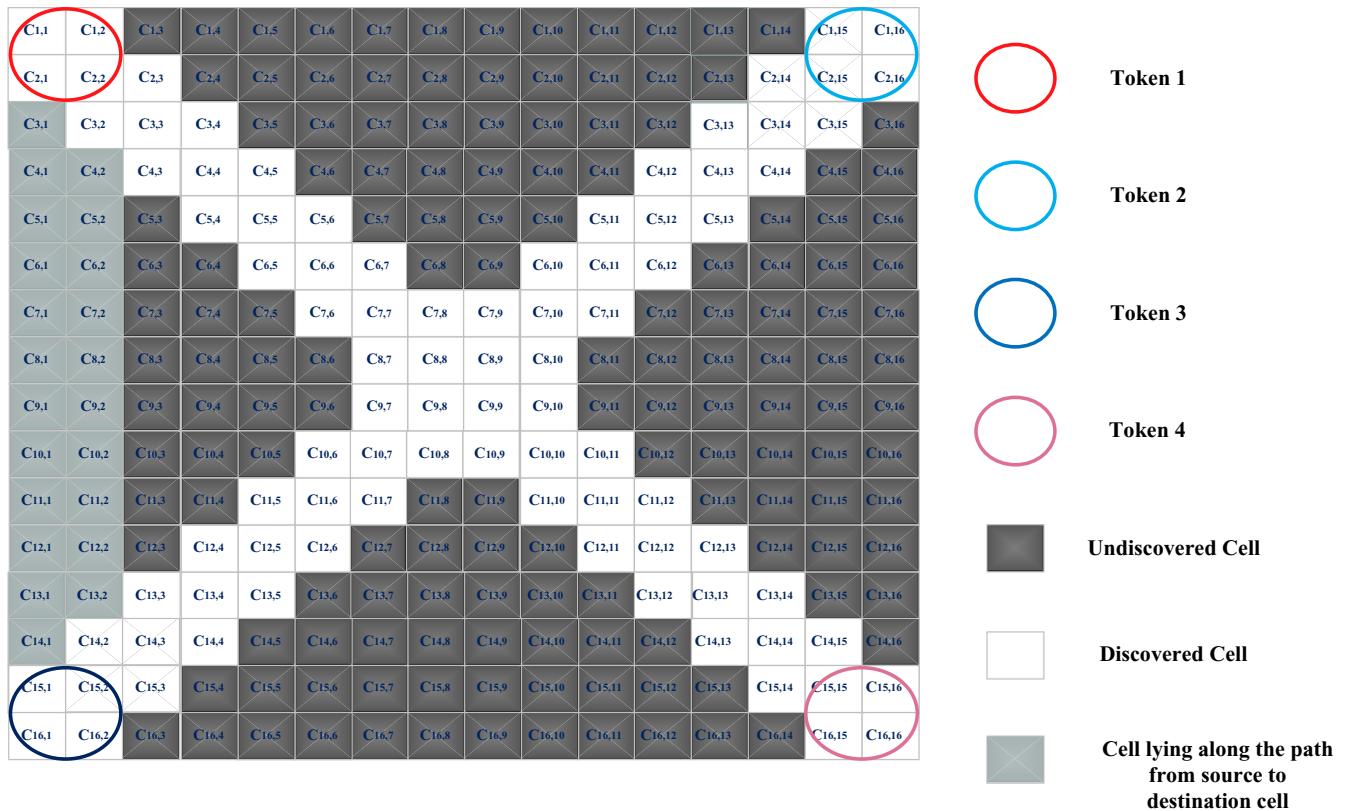


FIGURE 2 Token path planning with four tokens

The cost of reaching the probable farthest\_destination cell  $C_{m,n}$  for token  $t_k$  is obtained by taking the difference of  $d(CA_k, C_{m,n})$  and  $U(C_{m,n})$ , which is given by

$$(\text{Cost}_{m,n})_k = U(C_{m,n})_k - \beta \cdot d(CA_k, C_{m,n}) \quad (3)$$

where  $\beta$  is a constant. The cell  $C_{m,n}$  that has the maximum cost with respect to a token  $t_k$  is denoted as the farthest\_destination cell  $DC_k$  toward which the token  $t_k$  needs to traverse.  $DC_k$  is given by

$$DC_k = \max((\text{Cost}_{m,n})_k). \quad (4)$$

**TABLE 1** Description of parameters used for the token path planning algorithm

| Parameters                   | Description  |
|------------------------------|--|
| $a_C$                        | Unit area of a cell $C_{m,n}$  |
| $C_{m,n}$                    | Cell $C_{m,n}$ present in the $m$ th row and $n$ th column in map matrix $M$   |
| $CA_k$                       | Cell assigned to $k$ th token  |
| $Cost_{m,n}$                 | Cost of the cell $C_{m,n}$   |
| $CV(C_{m,n})$                | Color value representation for cell $C_{m,n}$ . WHITE represents cell discovered, BLACK represents cells undiscovered and GREY represents cell lying on the token's path between source and destination cell |
| $d(i, j)$                    | Euclidean distance between $i$ th and $j$ th cells   |
| $DC_k$                       | Destination cell assigned to $k$ th token  |
| $DR$                         | Region where sensor nodes are deployed   |
| $DU_{m,n}$                   | Distance utility value assigned based on distance of cell $C_{m,n}$ with respect to $k$ th token   |
| $M = [C_{m,n}]_{l \times b}$ | Map matrix of size $l \times b$ representing $R$ , $\forall C_{m,n}$ . Where $l$ and $b$ represents the length and breadth of the region $R$   |
| $n_b$                        | Number of unexplored or BLACK cells at any interval during token traversal   |
| $n_e$                        | Number of explored or WHITE cells at any interval during token traversal   |
| Path ( $s, d$ )              | Path traversed by token from source to destination cells   |
| $R$                          | Rectangular region where sensor deployed region $DR$ can fit in  |
| $R_k$                        | Communication range of $k$ th token holder.  |
| $T_k$                        | $k$ th token   |
| $TC_R$                       | Total number of cells in region $R$  |
| $UC_{m,n}$                   | Utility value assigned to the cell $C_{m,n}$ with respect to $k$ th token  |

It should be noted that the cells that figure in the path from the source cell to the farthest\_destination cells are marked as *GREY* to indicate that they will be explored in the near future and are considered equivalent to *WHITE* cells during computation of  $U(C_{m,n})$ . The main advantage of this is that it prevents other tokens from utilizing these cells in their path planning. The cells marked in *GREY* will be changed to *WHITE* once the token  $t_k$  has traversed through them. Once the first set of the farthest\_destination cells is determined for all tokens, the TPP algorithm repeats the procedure to determine the next set of the farthest\_destination cells. The TPP algorithm continues to repeat this mechanism until all cells in the entire region  $R$  have been marked *WHITE*.

**Algorithm 1** Max-gain algorithm

```

1: procedure TIME-GAIN( $M, R, a_C, R_k$ )
2:    $TC_R \leftarrow R/a_C$ 
3:   for all  $C_{m,n}$  do
4:     while  $n_e < TC_R$  do
5:       for all  $T_k$  do
6:         for all  $C_{m,n}$  do
7:           if  $CV(C_{m,n}) = BLACK$  then
8:              $n_b \leftarrow$  count of BLACK cells in
             Path( $CA_k, C_{m,n}$ )
9:             if  $d(CA_k, C_{m,n}) < R_k$  then
10:               $DU_{m,n} \leftarrow 1 -$ 
              ( $d(CA_k, C_{m,n})/R_k$ )
11:            else
12:               $DU_{m,n} \leftarrow 0$ 
13:            end if
14:             $UC_{m,n} \leftarrow \alpha \cdot n_b - (\sum_{i=1}^n DU_{m,n_i} -$ 
             $DU_{m,n_k})$ 
15:             $Cost_{m,n} \leftarrow UC_{m,n} -$ 
             $\beta \cdot dist(CA_k, C_{m,n})$ 
16:             $DC_k \leftarrow \max(Cost_{m,n}), \forall C_{m,n}$ 
17:          end if
18:        end for
19:        Mark Path( $CA_k, DC_k$ ) as GREY in  $M$ 
20:        Set destination of  $T_k$  to  $DC_k$ 
21:        Mark Path( $CA_k, DC_k$ ) as WHITE in
         $M$ , once  $T_k$  reaches  $DC_k$ 
22:      end for
23:    end while
24:  end for
25: end procedure

```

### 3.2 | Fetching the destination coordinates

Once the TPP algorithm is executed, the prospective destination coordinates for each token are obtained. These destination coordinates form the basis for the dissection of the entire deployment area into multiple regions, thereby initiating simultaneous neighbor discovery with the help of multiple tokens. Initially, the token holder node fetches the first set of destination points for the tokens to begin the neighbor discovery process. The number of coordinate points fetched depends on the number of tokens to be deployed in the network. When all coordinates of a path, obtained from the TPP algorithm, have been traversed, the tokens return to the original source. This mechanism, although mostly redundant, ensures that the nodes which had not previously been elected as token holders have a chance of being elected during the next phase of the process, thereby ensuring 100% node discovery. In the proposed algorithm, token movement is serialized within a bounded region, enabling the algorithm to scale-up to regions any size. Once the initial set of destination points are selected



and fixed, the process progresses to subsequent phases of neighbor discovery.

### 3.3 | Neighbor discovery by multiple tokens

In the proposed algorithm, token movement is restricted within a bounded region. Once an initial set of farthest\_destination cell coordinates is fetched by the token holder node, the proposed approach needs to calculate the bounded region, which in turn requires neighbor discovery. To begin with, the token holder node performs *Slow-Scan* to transmit *HELLO* packets to its neighboring nodes as depicted in Figure 1. During operation in the *Slow-Scan* mode, the token holder node switches slowly from one sector to another to ensure eventual coincidence of sectors with its neighboring nodes. This requires non-token holder nodes to be operating in *Fast-Scan* mode. On receiving the *HELLO* packet, each neighboring node responds with a *REPLY* packet. Upon the successful receipt of the *REPLY* packet, the token holder node makes an entry for the corresponding neighbor node in *neighbor table*. Whenever a node updates its neighbor table, it records the information in the form of 4-tuple of values following the template:  $\langle node\_id; X\_coordinate; Y\_coordinate; sector\_id \rangle$ .

It can be seen from Figure 1 that each neighbor discovery round is divided into  $\mu$  *Slow-Scan* periods, where  $\mu$  is the total number of sectors in a sectored antenna. Each *Slow-Scan* period is further subdivided into  $\mu$  *Fast-Scan* periods. The *Slow-Scan* and *Fast-Scan* periods indicate the staying period in a particular sector before switching to the next sector, for token holder nodes and non-token holder nodes, respectively. Furthermore, in each *Fast-Scan* period, the exchange of *HELLO-REPLY* packets takes place. The format of the *HELLO* packet is depicted in Figure 3A. Non-token holder nodes around the token holder operate in *Fast-Scan* mode. On receiving the *HELLO* packet, the nodes in *Fast-Scan* mode acknowledge it by sending a *REPLY* packet, which

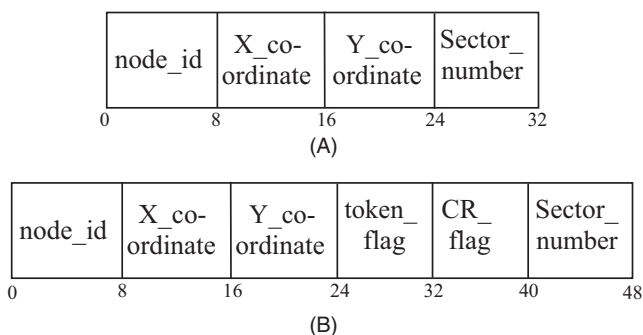
includes data about fields such as *node\_id*, *X\_coordinate*, *Y\_coordinate*, *token\_flag*, *CR\_flag*, and *sector\_number*, as depicted in Figure 3B. The *token\_flag* indicates whether or not a particular node has been a token holder during any previous phase. If it is unset, then the node has not previously been a token holder. The *CR\_flag* is used to resolve contentions, as explained in Section 3.6. Once the *REPLY* packet is received, the token holder node updates its neighbor table and selects the best candidate to pass the token to as per the *BetterTokenCandidate()* procedure described in Algorithm 4. The neighbor table records the *node\_id*, the location of the neighbor, the node's sector number in which the *REPLY* packet was received and the sector number of the corresponding neighbor from which the *REPLY* packet was transmitted. The token holder also checks whether or not the neighboring node that sent the *REPLY* packet lies in the intended direction of token traversal.

Once a candidate node is selected as the next bearer of the token within the bounded region, the token is passed to that node and this process is continued until the token reaches nodes that are closer to the destination coordinates. When no node is available in the intended direction of token traversal, a new unexplored destination will be fetched using Algorithm 3. The entire procedure is continued until all the predefined coordinates have been traversed by the tokens. The neighbor discovery process is carried out independently in each bounded region and simultaneously in any number of regions, as per the requirement.

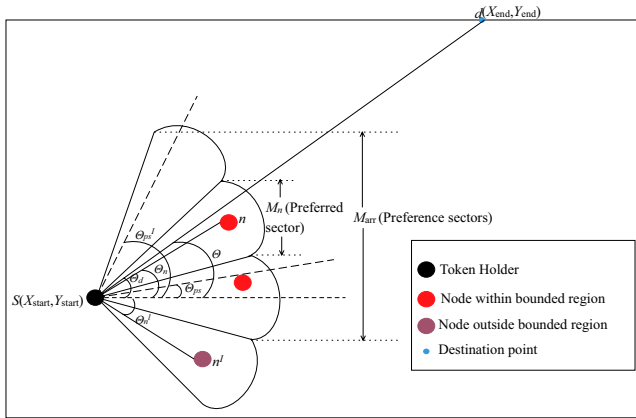
### 3.4 | Bounded region calculation

As the algorithm is centered around simultaneous neighbor discovery in multiple regions, confining the token passing process within separate bounded regions is of utmost importance. Thus, the choice of the candidate node to be the next token bearer must be restricted to the designated region corresponding to the token. The bounded region corresponding to each token includes the token traversal path from the source node to the destination node. The movement of the token following the above criteria is guided by the node-within-bounds (NWB) algorithm (Algorithm 2). Calculation of the bounded region is carried out whenever a new token holder is selected or whenever a new destination point is retrieved. This renews the entire process, thereby making it highly dynamic.

Figure 4 illustrates the token path calculation within a bounded region. In this figure, the BLACK node is the token holder, the RED node is the candidate for being the next token holder, which lies within the bounded region of the current token traversal path, and the node lying outside the bounded region has been colored PURPLE. We assume the source coordinates  $S(X_{start}, Y_{start})$  to be the same as the current position of the token holder node and the destination coordinates



**FIGURE 3** Packet format of (A) *HELLO* and (B) *REPLY* packets



**FIGURE 4** Selection of *Preference sectors* for token passing in directional sensor network

$d(X_{end}, Y_{end})$  to have been obtained from the preceding phase of the TPP algorithm. Within the restricted region, any token movement is permissible and such movement may lie within a single sector or over multiple sectors. In case of multiple sectors, the sector lying along the axis from the source coordinates  $S(X_{start}, Y_{start})$  to the destination coordinates  $d(X_{end}, Y_{end})$  is given maximal preference and is labeled to be the *preferred sector*, denoted by  $M_n$ , as shown in Figure 4. Initially, a single node holds all the tokens and fetches the requisite number of paths. At the beginning of the next phase,  $N$  tokens are relayed onto their respective regions, thereby initiating the neighbor discovery process for those nodes. The tokens are transmitted in the direction of the destination coordinates obtained from Algorithm 1. The angle between any two points  $(X_1, Y_1)$  and  $(X_2, Y_2)$  (Lines 5 and 6 of Algorithm 2) is computed as follows.

$$\theta = \tan^{-1} \frac{Y_2 - Y_1}{X_2 - X_1}. \quad (5)$$

As illustrated in the Figure 4, the angles between the token holder and the destination coordinates of the token ( $\theta_d$ ), the token holder and the node within the bounded region ( $\theta_n$ ), and the token holder and the node outside the bounded region ( $\theta_{n'}$ ) are all calculated using (5).

To relay the token to an immediate neighbor within the bounded region, the appropriate selection of sectors plays a vital role. A set of sectors called *Preference sectors*, ( $M_{arr}$ ), are selected to determine whether or not the candidate for the next token bearer lies within the bounded region. The bounded region is formed by a total of  $t$  sectors inclusive of the *preferred sector*  $M_n$  and the sectors adjacent to it. These sectors, depicted in the Figure 4, are determined by means of the following equation

$$t = \frac{s}{2} - 1 \quad (6)$$

where  $s$  is the number of sector antennas mounted on the node. Sectors that are farther away from the *preferred sector*  $M_n$  are given lesser preference when being considered for candidature with respect to token passing. Equation (6) is derived on the basis of the requirements that (a) The token travels in the desired direction, (b) In the absence of a node in the *preferred sector*, an alternate, less preferred sector should be chosen, and (c) Nodes in the deployed region should be discovered by the token without deviating from the predefined direction of traversal.

**Algorithm 2** Node-Within-Bounds algorithm

- 1: **procedure** NODEWITHINBOUNDS(currentTokenHolder,  $n$ )
- 2:      $(X_{start}, Y_{start}) \leftarrow$  Coordinates of current Token Holder node
- 3:      $(X_{end}, Y_{end}) \leftarrow$  Selected path coordinates from Max-gain TPP algorithm
- 4:      $(X_n, Y_n) \leftarrow$  Coordinate of node  $n$
- 5:      $\theta \leftarrow$  angle between  $(X_{start}, Y_{start})$  and  $(X_{end}, Y_{end})$
- 6:      $\theta_n \leftarrow$  angle between  $(X_{start}, Y_{start})$  and  $(X_n, Y_n)$
- 7:      $s \leftarrow$  Number of sectors
- 8:      $t$  gets Number of preference sectors obtained from (6)
- 9:      $M_n \leftarrow \lceil \frac{s\theta_n - \pi}{2\pi} \rceil$  where  $M$  is sector of  $\theta_n$
- 10:     $M_{arr} \leftarrow$  Array of sectors containing  $\theta_{ps}$  to  $\theta_{ps'}$  obtained from (7)–(8)
- 11:    **if**  $M_n$  is in  $M_{arr}$  **then**
- 12:      Return True
- 13:    **else**
- 14:      Return False
- 15:    **end if**
- 16: **end procedure**

Lines 9 and 10 assign a *preferred sector*  $M_n$  and *preference sectors*  $M_{arr}$  based on the angle  $\theta_n$ . The sectors adjacent to *preferred sector*  $M_n$ , that is,  $M_{arr}$ , are selected using the following equations:

$$\theta_{ps} = \theta - \left\lfloor \frac{t}{2} \right\rfloor \left( \frac{2\pi}{s} \right) \quad (7)$$

and

$$\theta_{ps'} = \theta + \left\lfloor \frac{t}{2} \right\rfloor \left( \frac{2\pi}{s} \right). \quad (8)$$

Equations (7) and (8) constitute the *preference sectors*  $M_{arr}$  which vary on the basis of the number of sector antennas

present. Based on whether or not the *preferred sector*  $M_n$  falls within the set of *preference sectors* specified by  $M_{arr}$ , the Algorithm 2 returns the value *TRUE* or *FALSE*, respectively (Lines 11–14), which is further used by the Algorithm 3 as explained below.

Once the tokens are relayed to the next token bearer node and no node is found in the direction given by (5), that is, in the direction defined by the angle  $\theta$  between the *source* and the *destination* coordinates, new *destination* coordinates ( $X_{end}$ ,  $Y_{end}$ ) are fetched from the TPP algorithm. During this phase, the *source* coordinates are set to be the coordinates of the current token holder node in the bounded region and the next set of destination coordinates is fetched using the *Change-Path-Direction* algorithm (Algorithm 3). The process is repeated until the tokens traverse all available TPP coordinates.

---

**Algorithm 3** Change-Path-Direction algorithm
 

---

```

1: procedure CHANGEDIRECTIONS( $r$ )
2:   nodeFoundFlag  $\leftarrow$  False

3:   for all nodes lying in coomunication range of current
       token holder in region  $r$  do
4:     if NODEWITHINBOUNDS( $this, n$ ) =
       True then
5:       nodeFoundFlag  $\leftarrow$  True
6:       break
7:     else
8:       continue
9:     end if
10:  end for

11:  if nodeFoundFlag = False then
12:     $X_{new} \leftarrow$   $X$  coordinate of unexplored region
       from Max-gain TPP algorithm
13:     $Y_{new} \leftarrow$   $Y$  coordinate of unexplored region
       from Max-gain TPP algorithm
14:  end if
15: end procedure

```

---

In Algorithm 3, Lines 3–10 depict the case in which a node is found in the bounded region. In such a case, the value of *nodeFoundFlag* is set to be *TRUE*. The token is subsequently passed to the best candidate selected using the *BetterTokenCandidate()* procedure (Algorithm 4). On the other hand, if no such node is found within the bounded region, then the value of *nodeFoundFlag* remains set to *FALSE*, and the next coordinates are fetched from the TPP algorithm and stored as new destination coordinates ( $X_{new}$ ,  $Y_{new}$ ) (Lines 11–14). Following that, the token holder computes the bounded region for token traversal using Algorithm 2 as explained above, based on the new set of coordinates ( $X_{new}$ ,  $Y_{new}$ ) obtained.

### 3.5 | Best candidate node selection

---

**Algorithm 4** Better-Token-Candidate algorithm
 

---

```

1: procedure BETTERTOKENCANDIDATE( $n_{old}, n_{new}$ )
2:   if  $n_{old}$  has been token holder but  $n_{new}$  has
       not been token holder then
3:      $n_{candidate} \leftarrow n_{new}$ 
4:   else if  $n_{new}$  has been token holder but  $n_{old}$ 
       has not been token holder then
5:      $n_{candidate} \leftarrow n_{old}$ 
6:   else if both  $n_{old}$  and  $n_{new}$  have been token holder or
       none of them are token holder then
7:      $\theta \leftarrow$  Angle between current token holder node
       and current destination coordinate from
       TPP algorithm

8:      $M_{arr} \leftarrow$  Array of preference sectors for angle  $\theta$ 

9:     Both  $n_{old}$  and  $n_{new}$  will be in some sector in  $M_{arr}$ 
       as they satisfy NODEWITHINBOUNDS( $n$ )
       procedure

10:    if  $n_{old}$  and  $n_{new}$  are in same sector then
11:      if  $RSSI$  of  $n_{new} > RSSI$  of  $n_{old}$  then
12:         $n_{candidate} \leftarrow n_{new}$ 
13:      else
14:         $n_{candidate} \leftarrow n_{old}$ 
15:      end if
16:    else
17:      if Sector of  $n_{new}$  is of higher preference
       than sector of  $n_{old}$  then
18:         $n_{candidate} \leftarrow n_{new}$ 
19:      else
20:         $n_{candidate} \leftarrow n_{old}$ 
21:      end if
22:    end if
23:  end if
24:  Return  $n_{candidate}$ 
25: end procedure

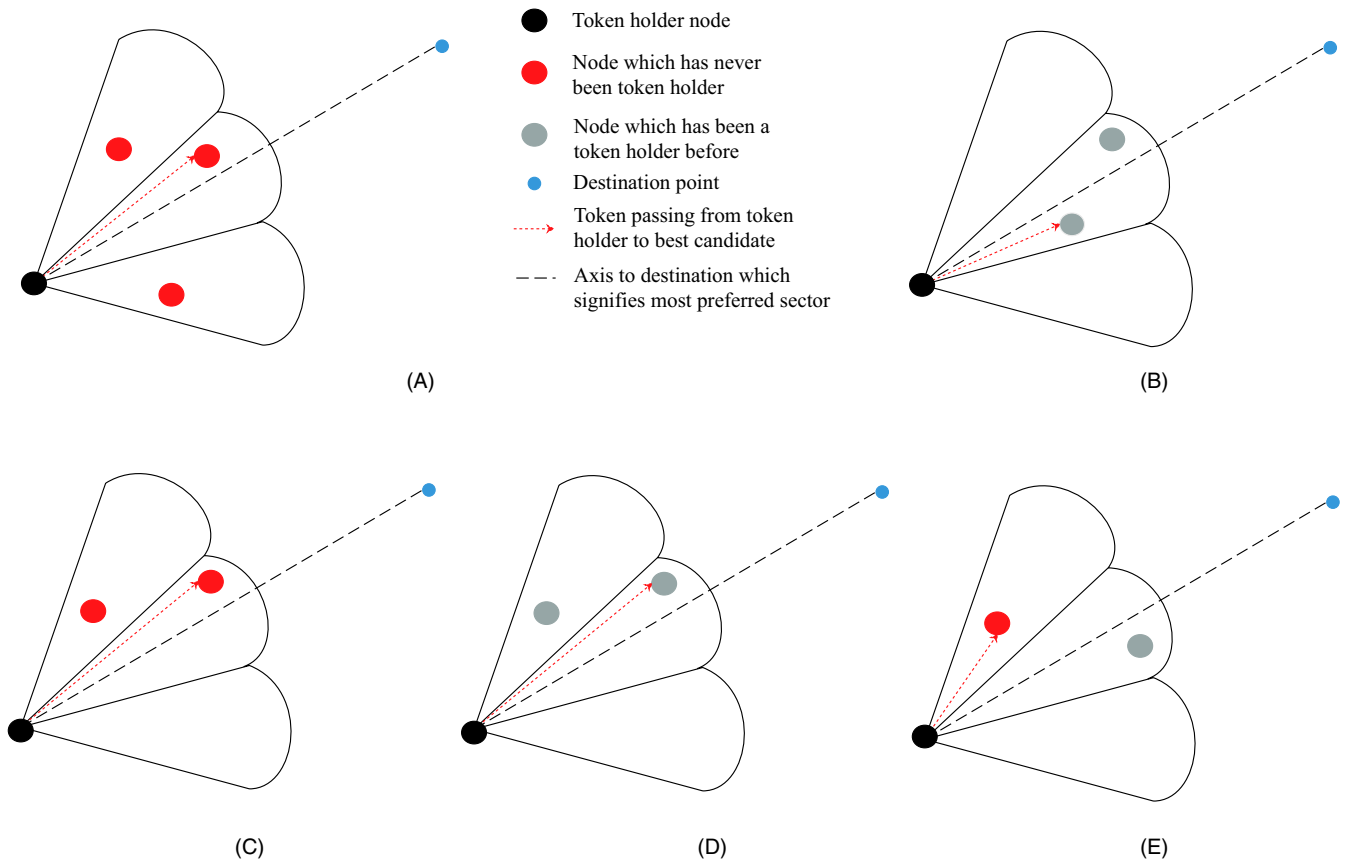
```

---

In the case in which there are two nodes in the bounded region, the decision regarding the selection of the next hop node is taken by the *BetterTokenCandidate()* procedure (Algorithm 4). The different scenarios in which the better candidate of the two nodes needs to be chosen as the next possible token holder are explained in Figure 5. If there are more than two nodes in the region, the algorithm is executed recursively to obtain the best candidate.

The *BetterTokenCandidate()* algorithm (Algorithm 4) is executed after restricting the nodes within the appropriate bounded region using Algorithm 2, thereby reducing the number of candidate token holders. Lines 2–5 give the highest preference to nodes which have not previously been a token holder. If such a node is found, then the algorithm returns  $n_{candidate}$ , which is the best





**FIGURE 5** Cases for selecting best candidate for token holder: (A) Neither nodes has previously been a token holder and they are in the same sector; (B) Both nodes have previously been token holders and are in the same sector; (C) Neither node has previously been token holder and they are in different sectors; (D) Both nodes have previously been token holders and are in different sectors; (E) One node has been a token holder and other node has not been token holder, where the nodes lie in different sectors

candidate node for becoming the next token holder. Lines 6–21 depict the case in which either both of the nodes have already been token holders, or neither of the nodes has been a token holder. In such a case, two scenarios may arise—the nodes lie in (a) the same sector or (b) different sectors. When the nodes lie in the same preference sector, the possible token holder candidate,  $n_{\text{candidate}}$ , is selected based on the received signal strength indicator (RSSI). The node with greater RSSI, that is, the node closer to the current token holder is selected (Lines 11–15), as illustrated in Figure 5. When the two nodes lie in different sectors, higher preference is accorded to the node that is in the sector that corresponds more closely with the angle  $\theta$  between the current token holder node and the destination coordinates (Lines 17–21). The best candidate node,  $n_{\text{candidate}}$ , is thus selected and the token is relayed to it.

### 3.6 | Token passing

The *Slow-Scan* operation, performed for  $\mu$  rounds, discovers all neighboring nodes and selects the candidate node to which the token is to be passed. Once the neighbor discovery phase ends, the token passing phase is carried out, during which the token

holder node activates the sector toward the selected candidate node for a *Slow-Scan* period and transmits the token. The candidate node receiving the token responds with the *REPLY* packet.

#### 3.6.1 | Contention resolution

As MuND uses multiple tokens, there is a possibility of contention between different tokens while the next token holder node is being selected. For example, suppose that nodes *A* and *B* are the current token holder nodes contending for the node *C* as the next prospective token holder node after running the *BetterTokenCandidate()* algorithm. Suppose, further, that the node *C* receives the token from both nodes *A* and *B* during the token passing phase and that it decides to keep token passed by node *A*. In this scenario, node *B* is unable to gather that node *C* has accepted node *A*'s token. Due to this missing information, node *B*'s token gets lost, thereby losing the path which *B*'s token was supposed to traverse.

To overcome such problems of contention, the prospective token holder node is programmed to send an acknowledgment in the form of *REPLY* packet, whose structure is given

in Figure 3B. The  $CR\_flag$  in the  $REPLY$  packet is used to indicate whether or not the prospective token holder node has already accepted a token. If the value of the  $CR\_flag$  is zero, it indicates that it has not yet become a token holder node. To relay the token, a token holder node (node  $A$  in our example) activates the sector where the prospective token holder node  $C$  lies, for a  $Slow-Scan$  period. On successful reception of the token, the prospective token holder node  $C$  acknowledges it by sending the  $REPLY$  packet. If  $CR\_flag$  is zero, it indicates that prospective token holder node is available to become the next token holder node. When node  $B$  transmits its token to node  $C$ , the latter replies with the value of  $CR\_flag$  set to 1, indicating that it has already accepted to be a token holder node corresponding to some other token. Following that, node  $B$  invokes the  $BetterTokenCandidate()$  algorithm once again to choose a new prospective token holder node.

## 4 | RESULTS

The performance of the proposed MuND protocol is evaluated using the Cooja simulator with directional antenna support [24]. For performance analysis, the proposed MuND protocol is compared with existing protocols like direct and indirect COND [19,22] and SAND [20] in terms of metrics like neighbor discovery latency, neighbor discovery ratio, control packet overhead and energy consumption. The simulation results are obtained by averaging results over multiple simulations. Table 2 depicts the Cooja simulator's configuration parameters.

The simulation experiments are carried out based on the following assumptions:

1. The static sensor nodes are uniformly and randomly deployed in a  $750 \times 750 \text{ m}^2$  area.
2. Each static sensor node is mounted with an antenna having eight sectors. Each individual sector antenna is assumed to be a rectangular patch antenna array (RPA) [25]. The RPA has a peak gain of 2.65 dBi, a beamwidth of  $45^\circ$  and a communication range of 135 m [25].
3. The sensor node receives beacon packets only if it lies inside the communication range of the sector antenna and if the receiver sensitivity is above  $-95 \text{ dBm}$ , which is the typical receiver sensitivity of CC2420 [26] radios used in WSN motes such as MicaZ [27].
4. In SAND and MuND protocols, the sector-switching interval for the token holder and non-token holder nodes are set at  $4\Delta t$  and  $32\Delta t$ , respectively.
5. In direct and indirect COND protocols, the fastest sector-switching interval is set to  $4\Delta t$  and slowest switching interval is set to  $32\Delta t$ .
6. For the proposed MuND protocol, four tokens are deployed in the network concurrently to initiate the process of neighbor discovery.

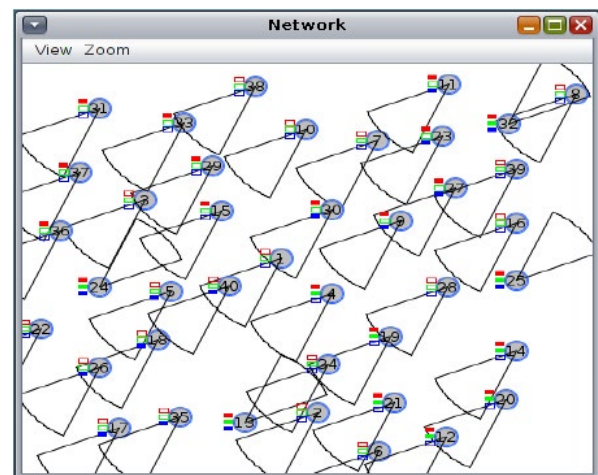
**TABLE 2** Configuration parameters for Cooja simulator

| Parameters                      | Description                  |
|---------------------------------|------------------------------|
| Deployment area                 | $750 \times 750 \text{ m}^2$ |
| Number of sensor nodes deployed | 20, 40, 60, 80               |
| Number of tokens used in MuND   | 4                            |
| Packet size                     | 128 bytes                    |
| Data rate                       | 250 Kbps                     |
| Radio frequency                 | 2.4 GHz                      |
| Number of sectors               | 8                            |
| Beamwidth of sector antenna     | $45^\circ$                   |
| Range of sector antenna         | 135 m                        |
| Receiver sensitivity            | $-95 \text{ dBm}$            |
| Transmission power              | 0 dBm                        |

The Figure 6 shows a screenshot of the MuND protocol, which performs neighbor discovery for the 40 deployed nodes using four tokens, namely  $t1$ ,  $t2$ ,  $t3$ , and  $t4$ . The undiscovered nodes have all their three LEDs set to OFF and the token holders have all their three LEDs set to ON. The nodes with red LED set to ON are the nodes that have been discovered by the token  $t1$ , the nodes with blue LED set to ON are the nodes that have been discovered by the token  $t2$ , the nodes with red and blue LEDs set to ON are the nodes that have been discovered by the token  $t3$ , and finally the nodes with red and green LEDs set to ON are the nodes that have been discovered by the token  $t4$ .

### 4.1 | Neighbor discovery ratio

Figure 7 depicts the neighbor discovery ratio, that is, the ratio of the number of nodes discovered to the total

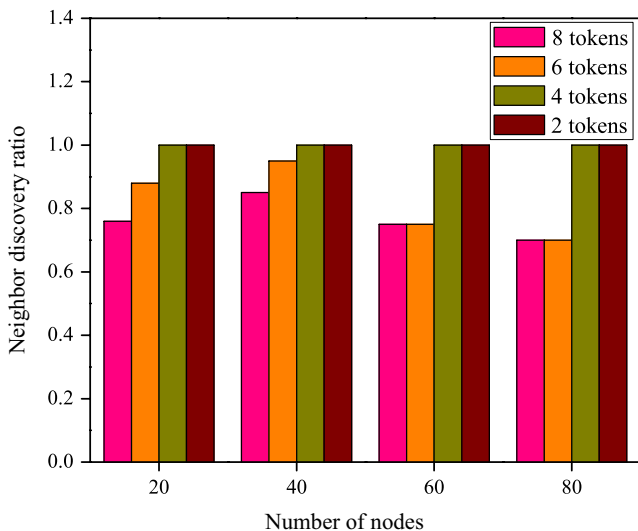


**FIGURE 6** Screenshot of multiple token-based neighbor discovery protocol for 40 deployed nodes

number of nodes deployed. The neighbor discovery ratio is calculated in networks with varying number of nodes deployed and varying number of tokens. In this simulation, the MuND protocol is executed with the number of tokens varying from four to eight. The use of four tokens for MuND not only ensures the least number of packet transmissions, but also guarantees 100% discovery of neighboring nodes. Using less than four tokens results in maximum discovery but at the cost of higher latency, while using more than four tokens exponentially increases the number of packets exchanged. This is mainly due to the increase in the number of token holders that transmit *HELLO* packets. Furthermore, in this case, as observed in Figure 7, neighbor discovery is not guaranteed to be 100% due to the fact that higher number of tokens implies higher number of token holders, which perform *Slow-Scan*. This may lead to the adjacent nodes simultaneously operating in *Slow-Scan* mode, and as a result, their sectors/beams may never intersect. This could cause them to miss out on discovering each other.

## 4.2 | Neighbor discovery latency

Figure 8 shows the neighbor discovery latency corresponding to the different numbers of nodes deployed. In all the scenarios depicted in Figure 8, the proposed MuND protocol completes 100% neighbor discovery with lower latency compared to that of all existing neighbor discovery protocols. Although direct COND and indirect COND initially performs at a faster rate, they fail to exceed discovery ratio of more than 96% on average in any of the aforementioned scenarios. Additionally, in direct and



**FIGURE 7** Neighbor discovery ratio of multiple token-based neighbor discovery protocol for varying number of tokens

indirect COND, once more than 90% of nodes have been discovered, the sector-switching interval remains constant for all the nodes. Hence, the beam intersection between discovered nodes and undiscovered nodes is fated to never occur. This further reduces the chances of discovering newer nodes.

## 4.3 | Control packet overhead

The comparison of control packet overheads for the neighbor discovery protocols being considered is depicted in Figure 9. As observed in the Figure 9, the proposed MuND protocol has a slightly higher control packet overhead compared to the SAND protocol. This is obvious, as multiple token holders exchange packets simultaneously in MuND to discover each other, whereas, in SAND protocol, only a single token holder initiates packet exchange with its neighbors. From Figure 9, it is clear that MuND protocol significantly outperforms direct COND and indirect COND protocols. As direct COND and indirect COND are unable to complete 100% neighbor discovery, as illustrated in Figure 8, we restrict the communication traffic until the SAND protocol completes neighbor discovery, so that these protocols have the chance to discover the remaining undiscovered nodes in this interval. The SAND protocol is chosen as the yardstick as it takes the longest duration for completion of the neighbor discovery process.

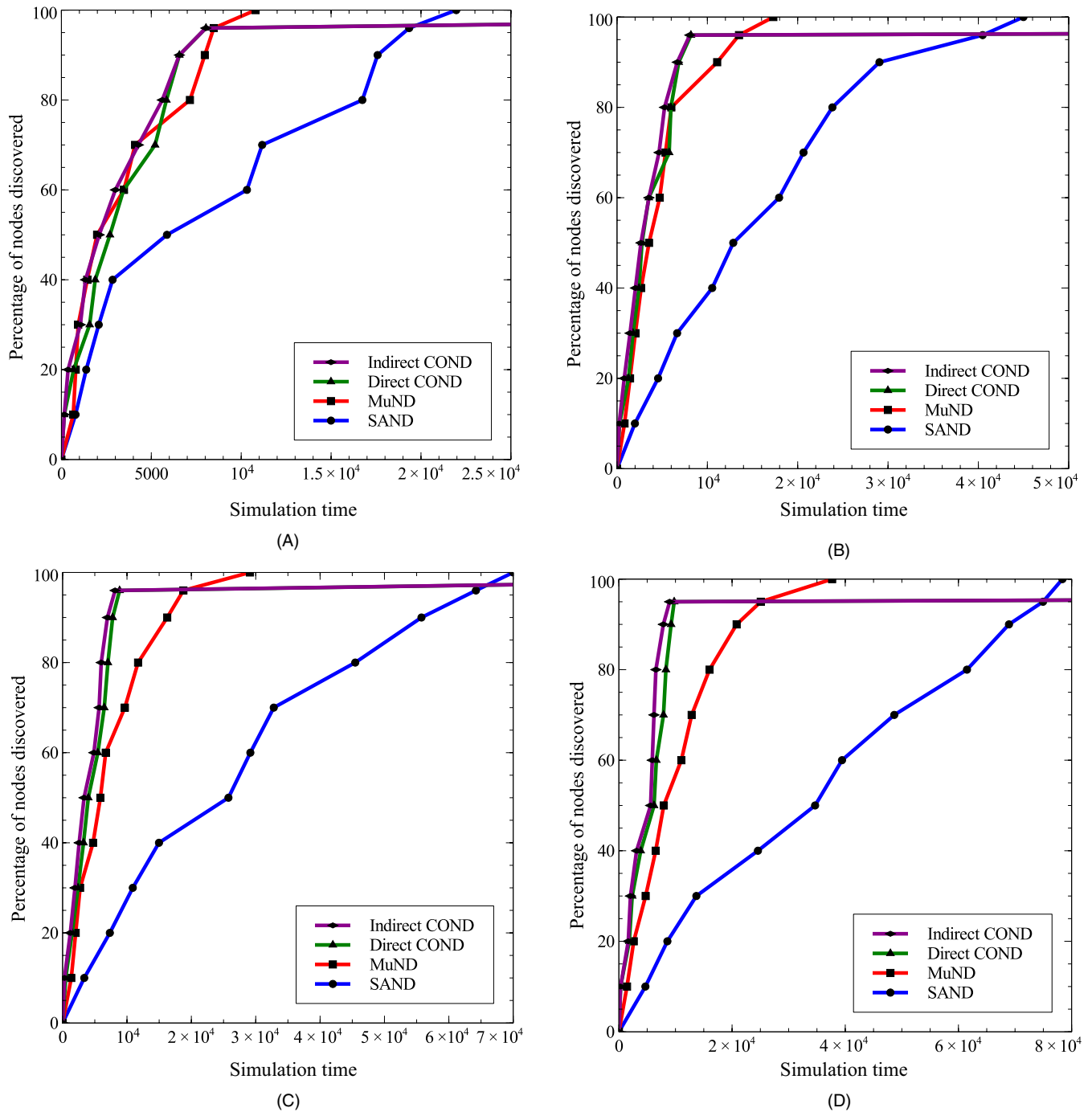
## 4.4 | Energy consumption

Energy consumption is another performance metric used to analyze the performance of neighbor discovery protocols. Since the deployed sensor nodes need to operate on a limited energy source, an energy efficient neighbor discovery protocol is preferable. In this subsection, the energy expenditures incurred during the communication of packets between neighboring nodes for the different protocols are studied. Energy consumption at a node occurs primarily due to the transmission and reception of *HELLO* packets, *REPLY* packets, and tokens.

Energy consumed by the sensor node is given by

$$Energy_{\text{sensor}} = Energy_{\text{recv}} [N_{\text{HELLO}} + N_{\text{Token}} + N_{\text{REPLY}}] + Energy_{\text{trans}} [N_{\text{HELLO}} + N_{\text{Token}} + N_{\text{REPLY}}] \quad (9)$$

where,  $N_{\text{HELLO}}$ ,  $N_{\text{Token}}$ , and  $N_{\text{REPLY}}$  denote the number of *HELLO* packets, tokens, and *REPLY* packets, respectively, and  $Energy_{\text{recv}}$  and  $Energy_{\text{trans}}$  denote the energy cost per packet for reception and transmission, respectively.  $Energy_{\text{recv}}$  is given by



**FIGURE 8** Neighbor discovery latency for varying number of nodes deployed: (A) 20 nodes; (B) 40 nodes; (C) 60 nodes; and (D) 80 nodes

$$Energy_{\text{recv}} = Power_{\text{recv}} \times \frac{Packet_{\text{size}}}{data_{\text{rate}}}, \quad (10)$$

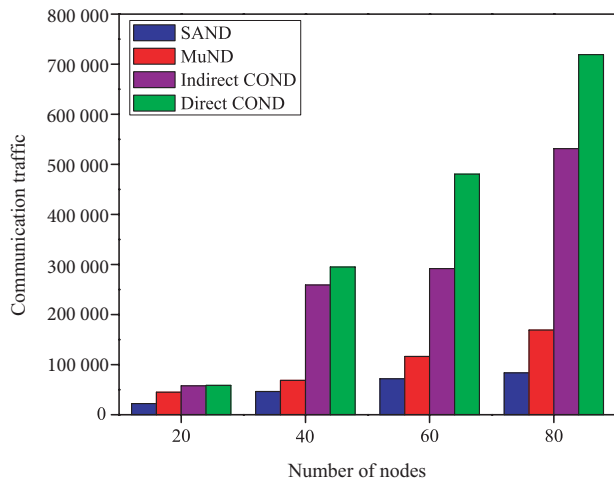
where  $Power_{\text{recv}}$  is the power consumed in receiving a packet.

$Energy_{\text{trans}}$  denotes the energy cost per packet for transmission and is given by

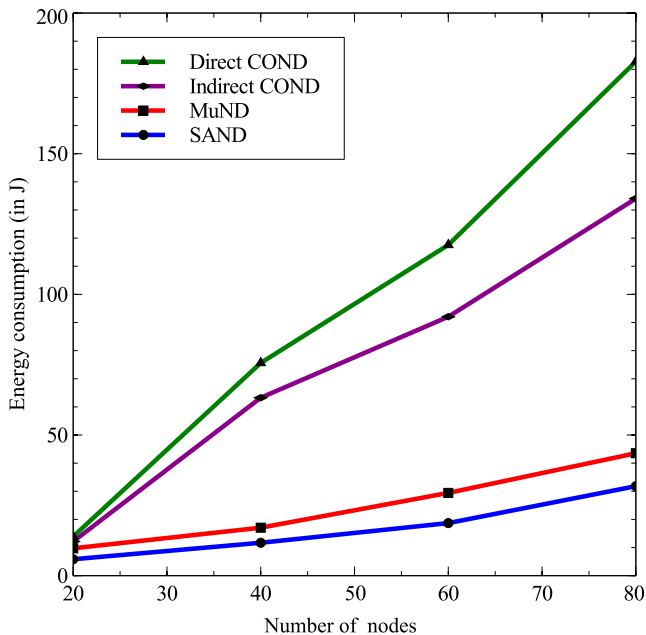
$$Energy_{\text{trans}} = Power_{\text{trans}} \times \frac{Packet_{\text{size}}}{data_{\text{rate}}} \quad (11)$$

where  $Power_{\text{trans}}$  is the power consumed for transmitting a beacon packet,  $Packet_{\text{size}}$  is the size of the packet, and  $data_{\text{rate}}$  denotes the data rate of the Zigbee protocol used in WSN.

According to CC2420 specifications [26], Micaz's transmitter and receiver modules consume currents of intensities 17.4 and 19.7 mA, respectively. The  $Packet_{\text{size}}$  of 128 bytes and the  $data_{\text{rate}}$  of 250 Kbps are considered. Based on (10) and (11), an  $Energy_{\text{recv}}$  of 0.242 mJ and an  $Energy_{\text{trans}}$  of 0.214 mJ are consumed during the receipt and transmission of a packet, respectively. The overall energy consumptions for



**FIGURE 9** Control packet overhead comparison for the neighbor discovery protocols



**FIGURE 10** Energy consumption comparison for the neighbor discovery protocols

communication by various neighbor discovery protocols are computed based on (9).

From Figure 10, it can be observed that COND protocols consume higher amounts of energy owing to higher control packet overheads, and the SAND protocol consumes lower overall energy at the cost of higher neighbor discovery latency. The proposed MuND protocol consumes slightly higher energy than the SAND protocol at the cost of lower neighbor discovery latency. Hence, there exists a trade-off between energy consumption and neighbor discovery latency. In WSN applications that assign primary importance to mission criticality, where the assigned task needs to be

carried out in the least possible time, the proposed MuND protocol is suitable.

## 5 | CONCLUSIONS

The proposed MuND protocol is a novel approach to tackle the latency and discovery ratio issues posed by other conventional neighbor discovery protocols. This protocol employs simultaneous MuND by dividing the entire deployment field into multiple bounded regions. The proposed protocol efficiently encompasses a number of tributary tasks like TPP, fetching destination coordinates, and bounded region calculation, which are combined to successfully solve the neighbor discovery problem in directional sensor networks. The performance analysis of the neighbor discovery protocols is carried out using the Cooja simulator with directional antenna support. The results show that the proposed MuND protocol has lower latency, higher neighbor discovery ratio, and relatively lower communication overhead and energy consumption than existing neighbor discovery protocols.

## ORCID

Shamanth Nagaraju  <https://orcid.org/0000-0001-9387-3414>

## REFERENCES

1. C. Santivanez and J. Redi, *On the use of directional antennas for sensor networks*, in Proc. IEEE Military Commun. Conf. MILCOM '03, Boston, MA, USA, Oct. 2003, pp. 670–675.
2. J. Jwa, *Tone dual-channel mac protocol with directional antennas for mobile ad hoc networks*, ETRI J. **34** (2012), no. 1, 98–101.
3. R. Ramanathan, *On the performance of ad hoc networks with beamforming antennas*, in Proc. ACM Int. Symp. Mobile hoc Netw. Comput., Long Beach, CA, USA, Oct. 2001, pp. 95–105.
4. R. Ramanathan et al., *Ad hoc networking with directional antennas: a complete system solution*, IEEE J. Sel. Areas Commun. **23** (2005), no. 3, 496–506.
5. H. Dai et al., *An overview of using directional antennas in wireless networks*, Int. J. Commun. Syst. **26** (2013), no. 4, 413–448.
6. B. El Khamlichi et al., *Collision-aware neighbor discovery with directional antennas*, in Proc. Int. Conf. Comput. Netw. Commun. (ICNC), Maui, HI, USA, Mar. 2018, pp. 220–225.
7. H. Li and Z. Xu, *Self-adaptive neighbor discovery in mobile ad hoc networks with directional antennas*, in Proc. IEEE Int. Conf. Comm. Workshops (ICC Workshops), Kansas City, MO, USA, May 2018, pp. 1–6.
8. M. Takai et al., *Directional virtual carrier sensing for directional antennas in mobile ad hoc networks*, in Proc. ACM Int. Symp. Mobile ad hoc Netw. Comput., Lausanne, Switzerland, June 2002, pp. 183–193.
9. T. Korakis, G. Jakllari, and L. Tassioulas, *CDR-MAC: a protocol for full exploitation of directional antennas in ad hoc wireless networks*, IEEE Trans. Mobile Comput. **7** (2008), no. 2, 145–155.
10. S. Vasudevan, J. Kurose, and D. Towsley, *On neighbor discovery in wireless networks with directional antennas*, in Proc. Annu. Joint Conf. IEEE Comput. Commun. Soc., Miami, FL, USA, Mar. 2005, pp. 2502–2512.



11. R. R. Choudhury et al., *Using directional antennas for medium access control in ad hoc networks*, in Proc. Annu. Int. Conf. Mobile Comput. Netw., Atlanta, GA, USA 2002, pp. 59–70.
12. R. R. Choudhury and N. H. Vaidya, *Deafness: a MAC problem in ad hoc networks when using directional antennas*, in Proc. IEEE Int. Conf. Netw. Protocols, Berlin, Germany, Oct. 2004, pp. 283–292.
13. R. Sharma et al., *A survey of mac layer protocols to avoid deafness in wireless networks using directional antenna*, Mobile Comput. Wireless Netw.: Concepts Methodol. Tools Appl., IGI Global, 2016, pp. 1758–1797.
14. S. Tehrani, A. F. Molisch, and G. Caire, *Directional zigzag: neighbor discovery with directional antennas*, in Proc. Global Commun. Conf. (GLOBECOM), San Diego, CA, USA, Dec. 2015, pp. 1–6.
15. L. Wei et al., *Lightning: a high-efficient neighbor discovery protocol for low duty cycle wsns*, IEEE Commun. Lett. **20** (2016), no. 5, 966–969.
16. D. Burghal, A. S. Tehrani, and A. F. Molisch, *On expected neighbor discovery time with prior information: Modeling, bounds and optimization*, IEEE Trans. Wireless Commun. **17** (2018), no. 1, 339–351.
17. Y. Wang, B. Liu, and L. Gui, *Adaptive scan-based asynchronous neighbor discovery in wireless networks using directional antennas*, in Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP), Hangzhou, China, Oct. 2013, pp. 1–6.
18. H. Cai and T. Wolf, *On 2-way neighbor discovery in wireless networks with directional antennas*, in Proc. IEEE Conf. Comp. Commun. (INFOCOM), Kowloon, Hong Kong, 2015, pp. 702–710.
19. F. N. Nur et al, *Collaborative neighbor discovery in directional wireless sensor networks*, Region 10 Conf. (TENCON), 2016, pp. 1097–1100.
20. R. Murawski et al., *Neighbor discovery in wireless networks with sectored antennas*, Ad Hoc Netw. **10** (2012), no. 1, 1–18.
21. Bo Liu et al., *Neighbor discovery algorithms in directional antenna based synchronous and asynchronous wireless ad hoc networks*, IEEE Wireless Commun. **20** (2013), no. 6, 106–112.
22. F. N. Nur et al., *Collaborative neighbor discovery in directional wireless sensor networks: algorithm and analysis*, EURASIP J. Wireless Commun. Netw. **2017** (2017), 1–15.
23. V. Sreejith et al., *A fast exploration technique in WSN for partition recovery using mobile nodes*, in Proc. National Conf. Commun. (NCC), Mumbai, India, 2015, pp. 1–6.
24. S. Nagaraju et al., *Realistic directional antenna suite for cooja simulator*, in Proc. National Conf. Commun. (NCC), Chennai, India, Mar. 2017, pp. 1–6.
25. S. Nagaraju et al., *Performance analysis of rectangular, triangular and E-shaped microstrip patch antenna arrays for wireless sensor networks*, in Proc. Int. Conf. Comput. Commun. Tech. (ICCCT), Allahabad, India, Sept. 2014, pp. 211–215.
26. CC2420 datasheet, Texas Instruments, available at <http://www.ti.com/lit/ds/symlink/cc2420.pdf>
27. MicaZ datasheet, Crossbow Corp., available at [http://www.mem-sic.com/userfiles/files/Datasheets/WSN/micaz\\_datasheet-t.pdf](http://www.mem-sic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf)

## AUTHOR BIOGRAPHIES



**Shamanth Nagaraju** is a PhD student in the Department of Computer Science and Information Systems at Birla Institute of Technology and Science (BITS) Pilani, K.K. Birla Goa Campus, Goa, India. He completed his MTech degree in Computer Network Engineering from Visvesvaraya Technological University, Karnataka, India in 2011. He obtained his BE degree in Computer Science Engineering from Vidya Vardhaka College of Engineering, Karnataka, India in 2009. His research interests include Wireless Sensor Networks, Localization and Smart antennas.



**Lucy J. Gudino** received her BE degree in Electronics and Communications Engineering from Kuvempu University, Karnataka, India, in 1994, and MTech and PhD degrees in Computer Science from Visvesvaraya Technological University, Karnataka, India, in 2003 and 2010, respectively. She is currently working as an Associate Professor in the department of Computer Science and Information Systems at Birla Institute of Technology and Science (BITS) Pilani, India. She has over 20 years of teaching experience in the field of Electronics and Communications, and Computer Science. Her research interests include Mobile communications, Adaptive Signal Processing, Adaptive arrays for cellular base stations, Wireless Networks (Wireless sensor networks and ad hoc networks) and Digital filter design and its application in Speech Processing.



**Nipun Sood** is pursuing Bachelor's degree in Department of Computer Science and Information Systems from Birla Institute of Technology and Science (BITS) Pilani, K.K. Birla Goa Campus, Goa, India. His research interests include Network Security and Cloud Computing.



**Jasmine G. Chandran** is pursuing Master's degree in Computer Science Engineering from Birla Institute of Technology and Science (BITS) Pilani, K.K. Birla Goa Campus, Goa, India. She obtained her BTech degree in Computer Science and Engineering from Government Engineering College Thrissur, Kerala, India in 2015. Her research interests include Wireless Sensor Networks, Internet of Things, and Distributed Systems.



**V. Sreejith** received the PhD degree in Computer Science from Birla Institute of Technology and Science (BITS) Pilani University, India. He is currently serving as a Lecturer in the Department of Computer Science and Information Systems, BITS Pilani K. K. Birla Goa Campus, Goa, India. He received his Bachelor's degree in Computer Science Engineering from Sun College of Engineering and Technology, under Manonmaniam Sundaranar University, Tamil Nadu, India in 2003. He completed his Master's degree in Computer Science Engineering in 2007 from Rajalakshmi Engineering College, under Anna University, Tamil Nadu, India. His research interests include Wireless Sensor Networks, Internet of things, and Cyber Physical Systems.