Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8425506, 11 pages
https://doi.org/10.1155/2022/8425506

Research Article

A Quasistraight Line Routing Protocol for Square Grid-Based Wireless Sensor Networks

Md. Ajij,[1] Sanjoy Pratihar,[2] Ashish Kumar Luhach,[3] and Diptendu Sinha Roy[1]

[1]*National Institute of Technology Meghalaya, Shillong, India*
[2]*Indian Institute of Information Technology Kalyani, Kalyani, India*
[3]*PNG University of Technology, Morobe, Papua New Guinea*

Correspondence should be addressed to Ashish Kumar Luhach; ashish.kumar@pnguot.ac.pg

Received 27 January 2022; Revised 18 March 2022; Accepted 23 March 2022; Published 11 May 2022

Academic Editor: Mohammad R Khosravi

Sensor nodes in a wireless sensor network (WSN) are both energy-constrained and vulnerable to faults and disasters. Communication between the sensor nodes is generally hop-by-hop, and the nodes are distributed throughout the area to be covered. Broadcast-based routing protocols are not preferable in sensor networks since broadcasting is considered costly in terms of battery power consumption. In this paper, a digital quasistraight line segment- (DQSS-) based approach is employed for the detection of quasistraight line segments, i.e., for quasistraight path finding between WSN sensors arranged in a square grid. Comparative results show that the method is comparable with the best-known straight line finding algorithm in terms of path lengths and computation time. Moreover, the proposed method is capable of avoiding dead nodes by updating DQSS parameters dynamically during path finding. Hence, the proposed method is promising to be used in WSN square grids as a quasistraight line routing protocol.

1. Introduction

Wireless sensor networks (WSNs) are extensively used in monitoring environments, surveillance equipments, intelligent home appliances operated remotely, patient care systems, etc. In WSN, the sensor nodes establish the path for communication from the sender to the receiver. This path making process should be carried out with limited resources. The performance of WSN is generally affected by many factors. These affecting factors are bandwidth, mobility, scalability, data aggregation, power consumption, etc. Because nodes have limited power sources, the minimization of power consumption is a vital issue in WSN, and this defines the performance of WSN [1]. Maximum energy is consumed by the sensor nodes in the communication processes. Routing protocols should be robust and straightforward, ensuring less energy consumption. Because of the limited resources of WSN nodes, the routing protocols must support the extended lifespan of the nodes [2]. Therefore, many protocols have been proposed highlighting the minimization of energy consumption.

Different protocols have been developed for WSNs according to the different prerequisites of uses and a large number of WSNs types. Numerous studies have attempted to analyze and classify these routing protocols according to different parameters that have been published. WSN routing protocol can be classified based on (a) application type, (b) delivery mode, (c) network architecture, (d) initiator of communication, (e) path establishment (route discovery), (f) network topology, (g) protocol operation, (h) next hop selection, and (i) latency-aware energy-efficient routing. The main goal of WSN is to establish a path consisting of the WSN nodes which will be reliable and energy efficient [3]. Energy consumption in routing is mainly due to finding

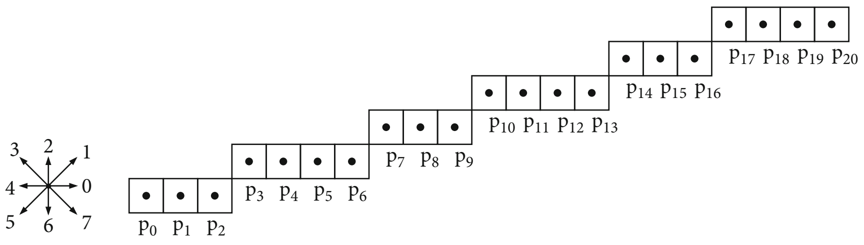

Figure 1: An example of a digital quasistraight line segment with singular code $s = 1$, nonsingular code $n = 0$, and the two run lengths (number of points in the run) l_1 and l_2 are 3 and 4, respectively.

neighbors for communications and necessary small computations. So, usually, the routing algorithms focus on how to compute the shortest path from the source node to the destination for quick transmission. A quick shortest path finding from the source node to sink may reduce the network congestion also.

Most of the traditional routing protocols cannot avoid the construction of curved (nonstraight) paths for data transmission. As a result, many multidirectional communications will lead to wastage of energy. Furthermore, the nonstraight paths normally contain more nodes than the straight paths, which leads to higher energy consumption [4]. So it is worthy of finding out a straight-line route (path).

Another serious problem is the recovery of failure nodes in the WSN environment. Node failure may be because of many reasons. The most crucial reasons for node failure in a wireless sensor network (WSN) [5] are (a) fabrication process problems, (b) environmental factors, (c) battery power depletion, and (d) enemy attacks. Node failure is a common issue in WSN, and this affects the connectivity in a network, which degrades the quality of communication [6]. In WSN, a connected network is desired for smooth communication. Hence, restoring connectivity is always given importance. Connectivity restoration is normally done by replacing dead nodes with other unused nodes [7]. This replacement mechanism should be robust, and the computational overhead must be taken care of as high-cost computations reduce the battery life [8–10]. WSNs and ad hoc networks are also vulnerable to faults, often disasters, and, owing to this very nature, are expected to fail and subsequently recover from such scenarios with minimal extraneous support [11, 12]. Energy optimal WSN operations have been studied extensively over the years, and the topologies and management strategies vary drastically with WSN use cases and applications [13, 14]. Various routing protocols have been studied, each with its own set of pros and cons. It has been well understood that traditional distributed routing involving broadcasts or those employing geographic information via GPS modules are not suited due to excessive battery drainage [4, 15–17]. This has paved the path for probabilistic routing such as gossip [18] and random routing [19, 20]. However, such probabilistic routing techniques are unsuited for WSNs with a considerable number of nodes as they cannot guarantee straight line paths, thus cannot ensure minimum distance, and are hence suboptimal in terms of energy expended while routing.

Table 1: Freeman's chain code for the line segment shown in Figure 1.

p_0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0

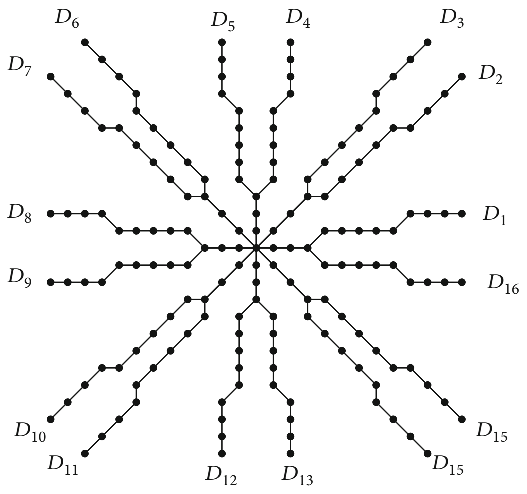

Figure 2: The sixteen directional DQSSs (representative segments with $l1 = 4$, $l2 = 5$).

1.1. Straight Line Routing. The random walk-based protocol is extensively used in WSN. Gossip [21] and rumor [20] are two well-known random walk-based routing protocols. Gossip concentrates on multicast, which suffers from power limitations and a high rate of wireless channel failure. In the rumor routing (RR) protocol, each node must maintain its list of neighbors. For propagation of a message, the node adds its list of neighbors to that message. Also, the message may keep track of all the nodes that this message has passed through. The node can decide a neighboring node to be the next node in the path. The next node must not have received the message earlier, and this way, it may prevent the route from growing in the backward direction. Rumor routing may show spiraling problems and energy is wasted in maintaining the records of visited nodes.

Chou et al. [22] proposed a routing protocol based on a random walk and straight line routing (SLR), intending to

Table 2: Direction understanding based on coordinates of the endpoints.

X-coordinate sign	Y-coordinate sign	$\lvert dy\rvert < \lvert dx\rvert$	$\lvert dy\rvert < \lvert \frac{dx}{2}\rvert$	$\lvert dx\rvert < \lvert \frac{dy}{2}\rvert$	No. of steps	Direction
Positive	Positive	Yes	Yes	No	$\lvert dy\rvert + 1$	D_1
Positive	Positive	Yes	No	No	$\lvert dx\rvert - \lvert dy\rvert + 1$	D_2
Positive	Positive	No	No	No	$\lvert dy\rvert - \lvert dx\rvert + 1$	D_3
Positive	Positive	No	No	Yes	$\lvert dx\rvert + 1$	D_4
Negative	Positive	No	No	Yes	$\lvert dx\rvert + 1$	D_5
Negative	Positive	No	No	No	$\lvert dy\rvert - \lvert dx\rvert + 1$	D_6
Negative	Positive	Yes	No	No	$\lvert dx\rvert - \lvert dy\rvert + 1$	D_7
Negative	Positive	Yes	Yes	No	$\lvert dy\rvert + 1$	D_8
Negative	Negative	Yes	Yes	No	$\lvert dy\rvert + 1$	D_9
Negative	Negative	Yes	No	No	$\lvert dx\rvert - \lvert dy\rvert + 1$	D_{10}
Negative	Negative	No	No	No	$\lvert dy\rvert - \lvert dx\rvert + 1$	D_{11}
Negative	Negative	No	No	Yes	$\lvert dx\rvert + 1$	D_{12}
Positive	Negative	No	No	Yes	$\lvert dx\rvert + 1$	D_{13}
Positive	Negative	No	No	No	$\lvert dy\rvert - \lvert dx\rvert + 1$	D_{14}
Positive	Negative	Yes	No	No	$\lvert dx\rvert - \lvert dy\rvert + 1$	D_{15}
Positive	Negative	Yes	Yes	No	$\lvert dy\rvert + 1$	D_{16}

```
Input: Geographical grid location of Source node (S) and Destination node (D)
Output: N: A quasistraight path from $S$ to $D$.
1 Translate the source node($(S(i,j))$from $(i,j)$ to $(0,0)$;
2 Translate the destination node $(D(u,v))$ accordingly;
3 Check x-coordinate sign, y-coordinate sign, if $|dy| < |dx|$, if $|dy| < |x/2|$, if $|dx|<|y/2|$;
4 Find the applicable row in Table 2 and get directional codes: s (singular code) and n (nonsingular code); See. Fig. 2
5 Find the number of steps $S_{count}$ and direction of DSS;
6 Break the $S_{count}$ in into two integers;
$S_{count} = k + m$;(where $0 \le |k| - |m| \le 1$);
7 Find the two run-lengths $l_1$ and $l_2$using the following criteria:
$S_{length} \longleftarrow \max(|dx|, |dy|)$;
$S_{length} \le k \times l_1 + m \times l_2$; Select $l_1$ and $l_2$when it shows minimum difference between $S_{length}$ and $(k \times l_1 + m \times l_2)$.
8 N=Find-Path$(l_1, l_2, s, n)$;
```

Algorithm 1: DQSS-based routing algorithm.

extend the route as straight as possible. The central idea of the SLR protocol was creating the routing path hop-by-hop. In each hop, the next node is selected so that it lies on the extended straight path approximately. Liu et al. [4] proposed a new protocol based on straight line routing. The rumor routing (RR) protocol also solves the spiral problem. The basic idea of discovering the straight path was to find the angle using radio signals. Many routing protocols are specially designed to enhance the classic RR protocol. For example, DRR [23], IDRR [24], SDRR [25], and ZRR [26] have been proposed to solve the spiral problem. Improved sensor node localization technique is proposed by Phoemphon et al. [27], where the positions of the anchor nodes form a straight or nearly straight line. Banimelhem et al. [28] proposed a principal component analysis- (PCA-) based efficient path generation algorithm.

1.2. Faulty Node.

Numerous strategies have been proposed for node deployment, which is often divided into two categories: random deployments and deterministic (grid-based) deployments [29]. Nodes are randomly eliminated and managed during ad hoc deportation in a stochastic deployment. When deploying on a grid, the nodes are arranged according to the angles of the grid points, which leads to greater accuracy in overall management. The physical

```
$p_{curr} \longleftarrow S$ ; $N \longleftarrow \varnothing$ ; Run⟵odd;index⟵1
while$p_{curr..x} \neq D.x$ A N D$p_{curr.y} \neq D.y$ do
    $N = N \cup p_{curr}$;
    if Run = odd then
        limit⟵$l_1$
    else
        limit⟵$l_2$
    while index ≤ limit do
        $p_{curr} \longleftarrow$ Point in the direction n from $p_{curr}$;
        $N = N \cup p_{curr}$;
        index⟵index+1
    if Run=odd then
        Run⟵even;
    if Run = even then
        Run⟵odd;
    $p_{curr} \longleftarrow$ Point in the direction s from $p_{curr}$;
    index⟵1;
if $p_{curr} \neq D$ then
Extend vertically or horizontally from $p_{curr}$ to D and condiser the points in N
return N;
```

PROCEDURE 1: Find path (l_1, l_2, s, n).

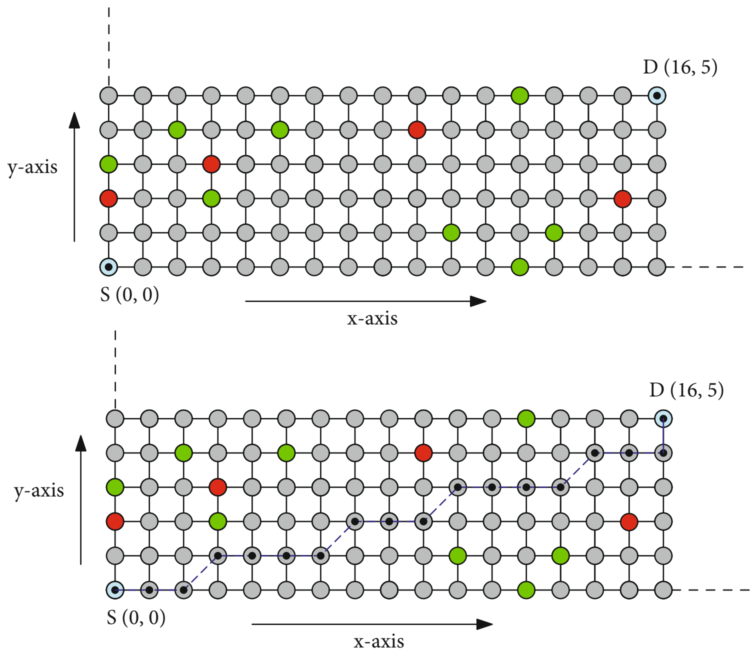

FIGURE 3: Demonstration of working of the algorithm; DQSS direction is $D1$, $n = 0$, $s = 1$, and $l_1, l_2 = 3, 4$; green nodes: active, gray nodes: sleeping, and red nodes: dead (the path does not go through any dead node).

positioning of sensor devices is better understood in grid-based deployment.

Many works exist to detect and analyze faulty nodes, and a few of them are listed below. Guo et al. [30] propose a sequence-based mechanism for detecting defective nodes. An algorithm for identification of fault node, based on a statistical z-score function, is proposed in [31], where all sensor nodes deliberately send information to the central node, and the root node analyzes the data to identify the fault. Asim et al. [32] provide an architecture for the management of faults in wireless sensor networks. They proposed that the entire network can be partitioned into the virtual lattice of cells and subsequently perform fault detection and recovery locally with the least energy utilization. A genetic algorithm-(GA-) based technique was proposed by Rajeswari and Neduncheliyan [33].

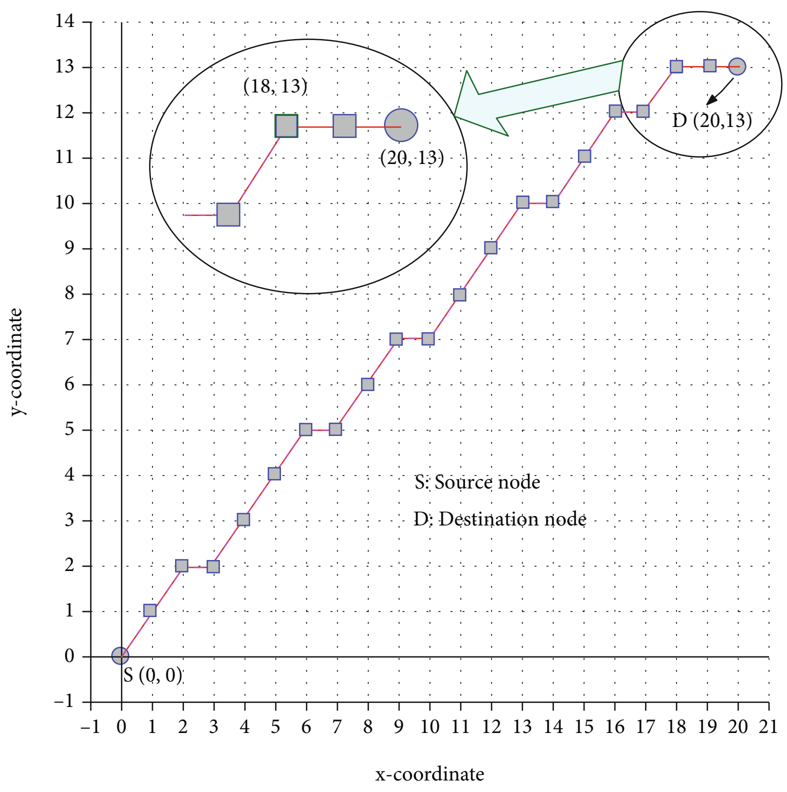

Figure 4: Demonstration of working of the algorithm. $n = 1$, $s = 0$, $l_1 = 3$, and $l_2 = 4$.

2. Our Contributions

In this work, we have proposed a novel path finding method based on quasistraight line fitting focusing on the grid-based deployment of sensor nodes. Moreover, we have proposed a protocol for path making and avoiding faulty nodes in a square grid of sensor nodes during path making. The protocol establishes a quasistraight line routing protocol for a node to node communication, involving a minimum possible number of sensors. We assume that there will be a few dead nodes in the sensor grid (mostly they are either live or sleeping). This proposed path making is fast and dynamic, and avoiding dead nodes does not incur extra communication costs.

3. Digital Quasistraight Line Segment (DQSS)

The structural view of rectangular grid-based wireless sensor network and points in digital space are indistinguishable. In our proposed work, our objective is to fit quasistraight digital line segments in the rectangular grid to find out the shortest path between two endpoints (source and destination) in WSN. The shortest distance between two points is indeed a straight line. In grid-based WSN, digital straight line will be suitable to explore the shortest route from sender to receiver.

Characterizations of digital straight lines have been given in many ways till date [34, 35]. Moreover, many algorithms exist to verify whether a given thin arc is digitally straight or not. Freeman introduced the chain code-based technique for representing 8 connected arcs and lines as a sequence of straight pieces [36, 37]. A chain code sequence (representing a digital curve) should possess the following properties if it represents a digital straight line segment (DSS) [34].

Table 3: Information stored at each sensor node.

Path ID	Run ID	Run limit	Node ID	n	s
$\{1, 2, 3, 4\}$	odd/even	l_1/l_2	in $\{1 \cdots l_1\}$ or $\{1 \cdots l_2\}$		

(i) (R1) The runs have at most two directions, differing by 45°, and for one of these directions, the run length must be 1

(ii) (R2) The runs can have only two lengths, which are consecutive integers

(iii) (R3) One of the runs can occur only once at a time

(iv) (R4) For the run length that occurs in runs, these runs can themselves have only two lengths, which are consecutive integers

In this proposed work, we characterize a straight line segment as the chain code sequence: $n^p s n^q s n^p \cdots$, where n is nonsingular code (the code occurs consecutively multiple times) and s is singular code (occurs singly in between nonsingular codes' runs). Code values, n and s, are consecutive integer differing by 45°. In our consideration, the nonsingular

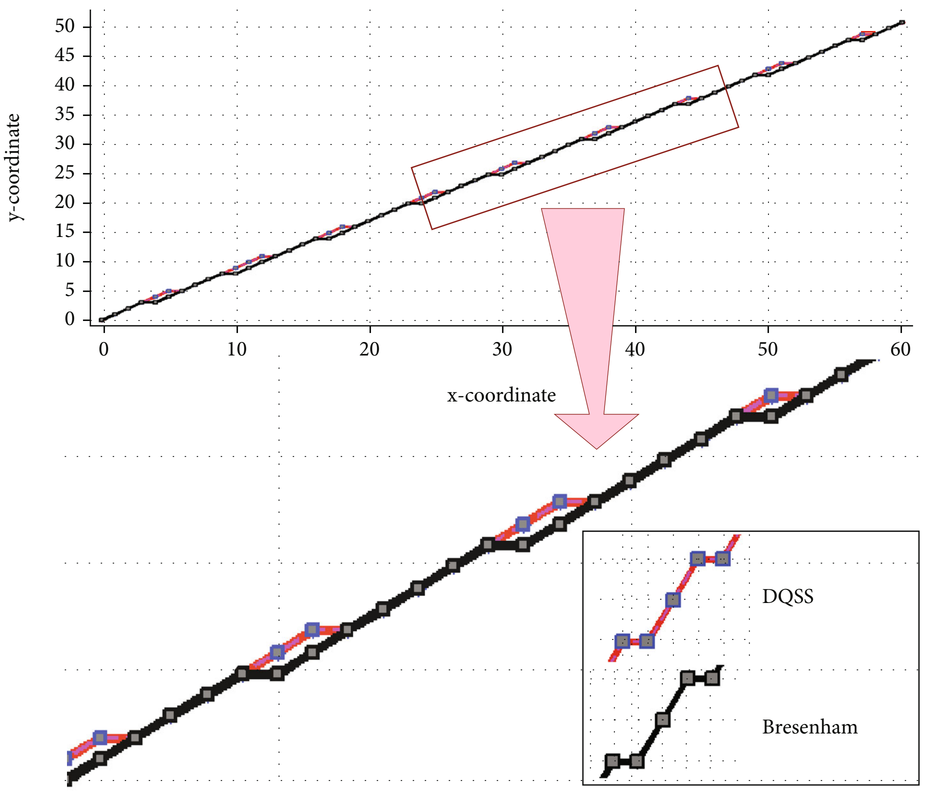

FIGURE 5: Differences in line segments: DQSS and Bresenham's line.

run lengths, p and q, are consecutive integers. Property $R1$ and $R2$ hold in our cases. On property $R3$, we are specific, because in our method, none of the runs occurs in runs. Instead, both the run lengths repeat alternately. So, it is evident that we may not always reach the destination point D. Whenever we reach the row or the column of the destination point in the grid, we use a horizontal or vertical stretch from that point to the destination point (D). Hence, we refer to the digital straight line segments obtained by us as digital quasistraight line segments (DQSS).

An example of a DQSS is shown in Figure 1, and the corresponding chain code is shown in Table 1. In Freeman's chain code-based properties, the mentioned run length refers to the length of a continuous sequence of the nonsingular code (in the chain code sequence). In our discussion, we have used it as the number of points in the continuous sequence in single direction. In the example shown in Figure 1, as per Freeman's definition, the two run lengths are $l_1 = 2$ and $l_2 = 3$, singular code $s = 1$, and nonsingular code $n = 0$. For the same example, we are considering the two run lengths $l_1 = 3$ and $l_2 = 4$.

3.1. Sixteen Directional DQSSs and Selection of DQSS. A digital quasistraight line segment will fall into one of the sixteen directional clusters as shown in Figure 2. Given the two endpoints $S(x_1, y_1)$ and $D(x_2, y_2)$ of a segment, the direction can be determined in Table 2. The singular and nonsingular codes concerning the various directions are as follows: $D1 : s = 1, n = 0$; $D2 : s = 0, n = 1$; $D3 : s = 2, n = 1$; $D4 : s = 1, n = 2$; $D5 : s = 3, n = 2$; $D6 : s = 2, n = 3$; $D7 : s = 4, n = 3$; $D8 : s = 3, n = 4$; $D9 : s = 5, n = 4$; $D10 : s = 4, n = 5$; $D11 : s = 6, n = 5$; $D12 : s = 5, n = 6$; $D13 : s = 7, n = 6$; $D14 : s = 6, n = 7$; $D15 : s = 0, n = 7$; and $D16 : s = 7, n = 0$. Next, we find the number of steps, S_{count}, in the straight line segment, using Table 2 and do the followings to estimate the run lengths l_1 and l_2.

(i) Break the number of steps, S_{count}, into two integers such that, $S_{\text{count}} = k + m$, where $0 \le |k| - |m| \le 1$

(ii) If the DQSS has run lengths, l_1 and l_2 then check the following criteria: $S_{\text{length}} \le k \times l_1 + m \times l_2$ and $S_{\text{length}} \longleftarrow \max(|\text{dx}|, |\text{dy}|)$

(iii) Select the DQSS which has minimum difference between S_{length} and $(k \times l_1 + m \times l_2)$

Our proposed DQSS-based quasistraight line finding method is shown in Algorithm 1. The algorithm selects the grid points or nodes to show the DQSS connectivity from the source node ($S_{i,j}$) to the destination node ($D_{u,v}$). The proposed algorithm selects and activates the nodes lying on the selected DQSS by maintaining the proper direction of the DQSS (following the properties as stated earlier), i.e., using the values l_1 and l_2 alternately starting from the source S and using the singular and nonsingular codes s and n as applicable. To start the path, we start with

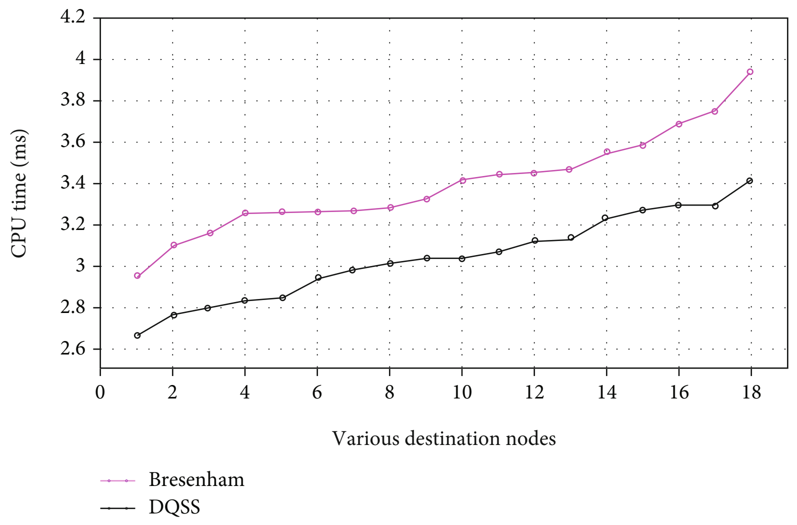

Figure 6: CPU time: DQSS vs. Bresenham's line (on 18 different line segments as shown in Table 4).

the smaller run length at source S. The selection of l_1, l_2, s, and n is shown in Algorithm 1 and the path making is shown in procedure FIND PATH (Procedure 1) of Algorithm 1.

4. Demonstration of the Proposed Algorithm

An example has been shown in Figure 3 to demonstrate the working of the proposed algorithm. Here in this example, the DQSS is to be fit in between $S(0,0)$ and $D(16,5)$. We find that the direction D_1 is applicable for this example. The number of stairs or steps, S_{count}, is $|\text{dy}|+1$, i.e., 6. We find the possible values of k and m as 3 and 3. As, S_{length} is 16 here, we find that l_1 and l_2 can be set as 3 and 4, respectively, following the criteria: $S_{\text{length}} \leq k \times l_1 + m \times l_2$. Hence, we start path finding from S using $n=0$, $s=1$, $l_1=3$, and $l_2=4$. As shown in the procedure of Algorithm 1, p_{curr} is the current point during path making. We extend from the current point using nonsingular code's run lengths as applicable. Stairs are created using the jumps because of the application of the singular code, and we gradually proceed towards the destination point D. If the current point p_{curr} reaches either the row (when $p_{\text{curr}}.y = D.y$) or the column (when $p_{\text{curr}}.x = D.x$) of the destination point in this process, we stretch horizontally or vertically towards D from that current point p_{curr}. An example has been shown in Figure 4. In this example, when p_{curr} reaches $(18,13)$, the y values of p_{curr} and D become equal. Hence, we stretch from $(18,13)$ to $(20,13)$.

5. The Protocol Using DQSS

Our proposed method works on a regular rectangular grid [38], where sensor nodes are positioned at grid intersection points. Our objective is to find the shortest quasistraight line path from the sender to the receiver. Sensor nodes are classified as given below:

(i) Active nodes: the nodes which are active in data transmissions

(ii) Sleeping nodes: initially, the nodes are sleeping and become activated based on requests

(iii) Dead nodes: dead nodes do not work in any condition as they are not in working condition; the dead nodes may be replaced with sleeping nodes

Wireless sensor nodes may be fixed nodes or mobile nodes. But in our case, we assume that the mobility of nodes is very less, and during movement, the nodes communicate with a core positioned at the grid points. So, virtually, the grid points are always the sensor nodes' locations. If a node is not active but lying on the detected straight line, then either it is a sleeping node or a dead node. If it is a sleeping node, the state of the node is changed from sleeping to active. If it is a dead node, then it does not respond to path making requests, and it is avoided reaching the destination. It must be noted that a dead node can be avoided by updating the run length limits, i.e., by preponing or postponing the application of the singular code. It is true that because of this preponing or postponing of the singular code, some runs may have run lengths other than l_1 or l_2. But, the length minimization constraint is maintained.

5.1. Sending and Receiving at Sensor Nodes. The starting point sensor initiates the path finding by sending a request to the prospective next sensor as per the codes and run lengths. We assume that the sensor nodes are equipped with local processors and storage registers to store their tagged information. If the next sensor responds to the previous, it is marked into the path, and the process continues until the destination is reached. The information which are tagged with each sensor are primarily path ID, run ID, run limit, node ID, n, and s as shown in Table 3. Here, path ID is the ID of the connecting straight line path. We assume that a sensor node can be part of four paths at most. Every path has several runs of codes. These runs are differentiated as

Table 4: Various line segments and the corresponding CPU time (in milliseconds).

	L1: dx = 10, dy = 6	L2: dx = 20, dy = 12	L3: dx = 30, dy = 18
DQSS	2.661	2.766	2.801
Bresenham	2.947	3.102	3.158
	L4: dx = 40, dy = 24	L5: dx = 50, dy = 30	L6: dx = 60, dy = 36
DQSS	2.833	2.843	2.941
Bresenham	3.250	3.260	3.259
	L7: dx = 70, dy = 42	L8: dx = 80, dy = 48	L9: dx = 90, dy = 54
DQSS	2.978	3.014	3.038
Bresenham	3.264	3.280	3.321
	L10: dx = 100, dy = 60	L11: dx = 120, dy = 72	L12: dx = 140, dy = 84
DQSS	3.039	3.063	3.119
Bresenham	3.406	3.440	3.446
	L13: dx = 160, dy = 96	L14: dx = 180, dy = 108	L15: dx = 200, dy = 120
DQSS	3.13	3.227	3.264
Bresenham	3.466	3.546	3.581
	L16: dx = 250, dy = 150	L17: dx = 300, dy = 180	L18: dx = 500, dy = 300
DQSS	3.288	3.288	3.409
Bresenham	3.682	3.748	3.939

odd and even. The initial run ID is odd, followed by an even ID run, followed by an odd ID run, and so on. If the run ID is odd, the sensor node is part of a run with length equal to l_1. If the run ID is even, the sensor node is part of a run with length equal to l_2. Node ID denotes the index of the node within the run. For example, if node ID is i and run ID is odd, then the current sensor is i-th node of a run whose length is l_1. Here, n and s are nonsingular and singular codes. All these pieces of information are stored and processed by the local processor. Using these values, we decide the next node at each sensor. For example, if the current node is l_1-th node in an odd run, then the next prospective node in the straight line lies in direction s from the current point. The current point is the latest point decided to be in the straight line segment.

For the DQSS example shown in Figure 3, the triplet (<RunID>,<RunLimit>,<NodeID >) values (at sensor nodes) starting from the source node are as follows: $(\text{odd}, 3, 1)$, $(\text{odd}, 3, 2)$, $(\text{odd}, 3, 3)$, $(\text{even}, 4, 1)$, $(\text{even}, 4, 2)$, $(\text{even}, 4, 3)$, $(\text{even}, 4, 4)$, $(\text{odd}, 3, 1)$, $(\text{odd}, 3, 2)$, $(\text{odd}, 3, 3)$, $(\text{even}, 4, 1)$, $(\text{even}, 4, 2)$, $(\text{even}, 4, 3)$, $(\text{even}, 4, 4)$, $(\text{odd}, 3, 1)$, $(\text{odd}, 3, 2)$, and $(\text{odd}, 3, 3)$. Whenever a sensor node gets the node ID equal to run limit, it flips the run ID (odd to even or even to odd) and selects the next node in direction s from the current point.

The decision-making process is very lightweight as no other arithmetic or complex computations are involved. The nodes check the index limits at every node and forward the incremented index information, whereas in Bresenham's line drawing algorithm, we need to compute the decision parameter's value at each pixel position to decide the next pixel [39]. In Bresenham's algorithm, the decision parameter is updated by involving addition, multiplication, and comparison operations. In contrast, our method computes the necessary parameters once and only uses increment and comparison operations in the loop. Figure 5 shows a comparison of the DQSS line with the Bresenham's line. Also, a comparison between DQSS and Bresenham's line algorithm on the CPU times for various line segments is shown in Figure 6 and Table 4.

5.2. Dead Node Avoidance. We assume that significantly fewer dead nodes will be present in the grid. During the making of the straight line path, at some point, if a dead node appears as the following selection, we wish to avoid it. This is done by increasing or decreasing the current nonsingular run length (run limit). We have the following two cases.

(i) The current node p_{curr} is a dead node, and it is the first node of a run (reached using singular code s from the previous point). The run limit of the current run is increased by 1. An example is shown in Figure 7

(ii) The current node p_{curr} is a dead node, and it is any node other than the first node of a run (reached using nonsingular code n from the previous point). The run limit of the current run is reset by index − 1 (index points the current node p_{curr}. Hence, p_{curr} is avoided by applying a move using the singular code on the previous node of p_{curr}. An example is shown in Figure 8

5.3. Energy Consumption. In WSN, the energy consumed is the sum total of energy consumed by individual nodes (see

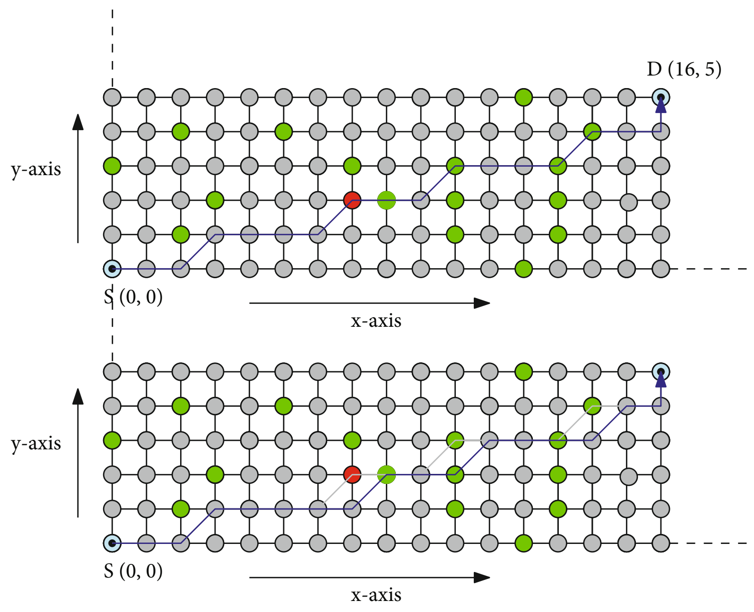

Figure 7: Dead node avoidance by increasing the current run length limit; the length of the first even run is increased by 1.

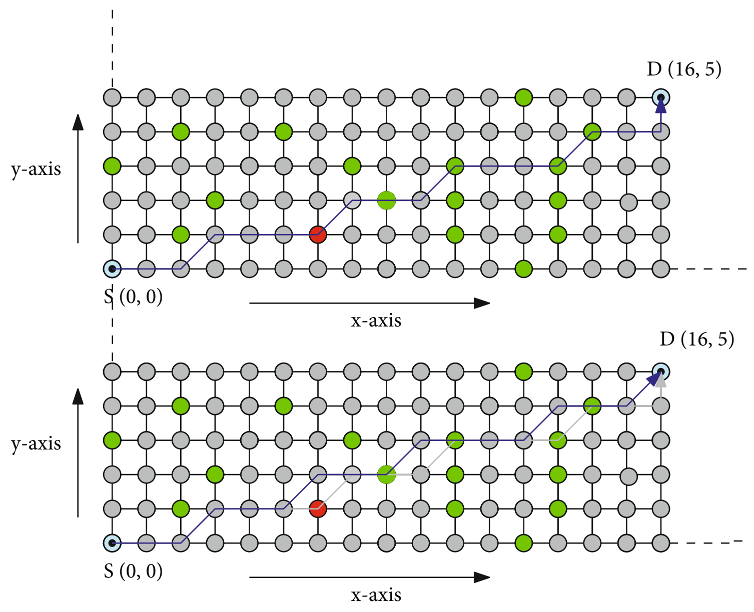

Figure 8: Dead node avoidance by decreasing the current run length limit; limit is reduced to 3 from 4.

Equation (1)) [40, 41]. Energy consumed by a node comprises of energy for transmitting packets (E_t), that for receiving packets (E_r), and consumptions because of sleeping (E_s).

$$E_{\text{Total}} = \sum_{i=1}^{n} E_i, \tag{1}$$

where $E_i = E_t + E_r + E_s$.

For E_r and E_s, most of the network simulators use standard values. However, E_t depends on various factors. Most prominent of which includes packet size (l) and distance between nodes (d). Hence, E_t may be expressed using the formula shown in

$$E_t = l * E_{\text{bit}} + l * \lambda * d^2, \tag{2}$$

where λ is medium constant.

Our proposed algorithm focuses on minimizing the path length between the two given nodes by finding a quasistraight line segment between the two nodes. Minimization of the path length ensures minimization of the energy consumption.

6. Conclusion

This paper proposes a novel quasistraight line routing protocol based on quasistraight line fitting, which is derived from Freeman's chain code-based straightness properties. The proposed algorithm focuses on the grid-based deployment of sensor nodes in WSN. If the constructed path attempts to go through a dead node, the path is modified so that the length minimization constraint is maintained with minimum deviation. The method has been compared with a standard straight line finding algorithm, and the results show its applicability.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] F. Bouakkaz and M. Derdour, "Maximizing WSN life using power efficient grid-chain routing protocol (PEGCP)," *Wireless Personal Communications*, vol. 117, no. 2, pp. 1007–1023, 2021.

[2] R. Zagrouba and A. Kardi, "Comparative study of energy efficient routing techniques in wireless sensor networks," *Information*, vol. 12, no. 1, p. 42, 2021.

[3] J. Yan, M. Zhou, and Z. Ding, "Recent advances in energy-efficient routing protocols for wireless sensor networks: a review," *IEEE Access*, vol. 4, pp. 5673–5686, 2016.

[4] H.-H. Liu, S. Jia-Jang, and C.-F. Chou, "On energy-efficient straight-line routing protocol for wireless sensor networks," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2374–2382, 2017.

[5] S. Umamaheswari, W. S. Antony, and A. Joe, "Detection and correction of node failures in wireless sensor networks," in *International Conference on Advanced Computing and Communication Systems (ICACCS), volume 1*, pp. 1479–1483, Coimbatore, India, 2021.

[6] R. N. Jadoon, A. A. Awan, M. A. Khan, W. Zhou, and A. Shahzad, "An Efficient Nodes Failure Recovery Management Algorithm for Mobile Sensor Networks," *Mathematical Problems in Engineering*, vol. 2020, Article ID 1749467, p. 14, 2020.

[7] K. Mahmood, M. K. Saeed, S. Ali, S. Zaman, A. Al Awady, and M. Saqib, "Smart node relocation (snr) and connectivity restoration mechanism for wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, pp. 1–19, 2021.

[8] H. Yetgin, K. T. K. Cheung, M. el-Hajjar, and L. Hanzo, "A survey of network lifetime maximization techniques in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 828–854, 2017.

[9] A. A. Aziz, Y. Ahmet Sekercioglu, P. Fitzpatrick, and M. Ivanovich, "A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 121–144, 2013.

[10] F. Engmann, F. A. Katsriku, J.-D. Abdulai, K. S. Adu-Manu, and F. K. Banaseka, "Prolonging the lifetime of wireless sensor networks: a review of current techniques," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 8035065, 23 pages, 2018.

[11] I. Benkhelifa, N. Nouali-Taboudjemat, and S. Moussaoui, "Disaster management projects using wireless sensor networks: an overview," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 605–610, Victoria, BC, Canada, 2014.

[12] D. G. Reina, M. Askalani, S. L. Toral, F. Barrero, E. Asimakopoulou, and N. Bessis, "A survey on multihop ad hoc networks for disaster response scenarios," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, Article ID 647037, 2015.

[13] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[14] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[15] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.

[16] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, New Orleans, LA, USA, 1999.

[17] R. S. Battula and O. S. Khanna, "Geographic routing protocols for wireless sensor networks: a review," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 12, pp. 39–42, 2013.

[18] E. Ahvar, S. Ahvar, G. M. Lee, and N. Crespi, "An energy-aware routing protocol for query-based applications in wireless sensor networks," *The Scientific World Journal*, vol. 2014, Article ID 359897, 9 pages, 2014.

[19] B. Blywis, M. Güneş, F. Juraschek, and S. Hofmann, "Gossip routing in wireless mesh networks," in *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1572–1577, Istanbul, Turkey, 2010.

[20] D. Braginsky and D. Estrin, "Rumor routing algorthim for sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 22–31, Atlanta, Georgia, USA, 2002.

[21] M.-J. Lin, K. Marzullo, and S. Masini, *Gossip Versus Deterministic Flooding: Low Message Overhead and High Reliability for Broadcasting on Small Networks*, Technical report, Department of Computer Science and Engineering, University of California, San Diego, USA, 1999.

[22] C.-F. Chou, S. Jia-Jang, and C.-Y. Chen, "Straight line routing for wireless sensor networks," in *10th IEEE Symposium on Computers and Communications (ISCC'05)*, pp. 110–115, Murcia, Spain, 2005.

[23] H. Shokrzadeh, A. T. Haghighat, F. Tashtarian, and A. Nayebi, "Directional rumor routing in wireless sensor networks," in *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*, pp. 1–5, Tashkent, Uzbekistan, 2007.

[24] H. Shokrzadeh, M. Mashaiekhi, and A. Nayebi, "Improving directional rumor routing in wireless sensor networks," in

2007 Innovations in Information Technologies (IIT), pp. 108–112, Dubai, United Arab Emirates, 2007.

[25] S. Hamid, A. M. Rahmani, A. T. Haghighat, and N. Forouzideh, “SDRR: serial directional rumor routing in wireless sensor networks,” in *2010 International Conference on Networking and Information Technology*, pp. 75–79, Manila, Philippines, 2010.

[26] T. Banka, G. Tandon, and A. P. Jayasumana, “Zonal rumor routing for wireless sensor networks,” in *International Conference on Information Technology: Coding and Computing (ITCC’05)-Volume II, volume 2*, pp. 562–567, Las Vegas, NV, USA, 2005.

[27] S. Phoemphon, C. So-In, and N. Leelathakul, “Improved distance estimation with node selection localization and particle swarm optimization for obstacle-aware wireless sensor networks,” *Expert Systems with Applications*, vol. 175, article 114773, 2021.

[28] O. Banimelhem, E. Taqieddin, and I. Shatnawi, “An efficient path generation algorithm using principle component analysis for mobile sinks in wireless sensor networks,” *Journal of Sensor and Actuator Networks*, vol. 10, no. 4, p. 69, 2021.

[29] M. A. Fadi, E. A. Ashraf, S. H. Hossam, and A. I. Mohamed, “Deploying faulttolerant grid-based wireless sensor networks for environmental applications,” in *IEEE Local Computer Network Conference*, pp. 715–722, Denver, CO, USA, 2010.

[30] S. Guo, Z. Zhong, and T. He, “Find: faulty node detection for wireless sensor networks,” in *Proceedings of the 7th ACM conference on embedded networked sensor systems*, pp. 253–266, Berkeley, California, 2009.

[31] R. R. Panda, B. S. Gouda, and T. Panigrahi, “Efficient fault node detection algorithm for wireless sensor networks,” in *2014 International Conference on High Performance Computing and Applications (ICHPCA)*, pp. 1–5, Bhubaneswar, India, 2014.

[32] M. Asim, H. Mokhtar, and M. Merabti, “A fault management architecture for wireless sensor network,” in *2008 International Wireless Communications and Mobile Computing Conference*, pp. 779–785, Crete, Greece, 2008.

[33] K. Rajeswari and S. Neduncheliyan, “Genetic algorithm based fault tolerant clustering in wireless sensor network,” *IET Communications*, vol. 11, no. 12, pp. 1927–1932, 2017.

[34] A. Rosenfeld, “Digital straight line segments,” *IEEE Transactions on Computers*, vol. C-23, no. 12, pp. 1264–1269, 1974.

[35] P. Bhowmick and B. B. Bhattacharya, “Fast polygonal approximation of digital curves using relaxed straightness properties,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590–1602, 2007.

[36] H. Freeman, “On the encoding of arbitrary geometric configurations,” *IRE Transactions on Electronic Computers*, vol. EC-10, no. 2, pp. 260–268, 1961.

[37] H. Freeman and L. S. Davis, “A corner-finding algorithm for chain-coded curves,” *IEEE Transactions on Computers*, vol. -C-26, no. 3, pp. 297–303, 1977.

[38] G. Ramamurthy, T. JagannadhaSwamy, and A. Jain, “Cost and energy efficient distributed computation: wireless sensor networks on uniform grid,” in *2021 International Conference on Sustainable Energy and Future Electric Transportation (SEFET)*, pp. 1–5, Hyderabad, India, 2021.

[39] J. E. Bresenham, “Algorithm for computer control of a digital plotter,” *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[40] T. D. Nguyen, J. Y. Khan, and D. T. Ngo, “A distributed energy-harvesting-aware routing algorithm for heterogeneous IoT networks,” *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 1115–1127, 2018.

[41] S. Verma, Y. Kawamoto, and N. Kato, “Energy-efficient group paging mechanism for qos constrained mobile IoT devices over LTE-A Pro networks under 5G,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9187–9199, 2019.