Received November 2, 2020, accepted November 5, 2020, date of publication November 16, 2020, date of current version November 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038047

Flood Control Distance Vector-Hop (FCDV-Hop) Localization in Wireless Sensor Networks

AZYYATI ADIAH ZAZALI, (Fellow, IEEE), SHAMALA K. SUBRAMANIAM, (Member, IEEE), AND ZURIATI AHMAD ZUKARNAIN, (Member, IEEE)
Department of Communication Technology and Networks, Universiti Putra Malaysia, Selangor 43400, Malaysia

Corresponding author: Azyyati Adiah Zazali (azyyatiadiah@gmail.com)

This work was supported by Universiti Putra Malaysia Publication Grant.

ABSTRACT Distance Vector-Hop (DV-Hop) localization is a distributed, hop by hop positioning algorithm. Anchor nodes generate the packets of nodes position information to calculate the number of hops between the beacon nodes and the anchor nodes, and these packets are flooded in the respective Wireless Sensor Networks (WSN). The DV-Hop algorithm obtains the distance information from the unknown nodes to the anchor nodes through the network topology calculation rather than the radio wave signals measurement which are used in the range-based algorithm. However, flooding information from all anchors to all beacon nodes will become too expensive for large networks (i.e., cost, energy), even with low anchor fraction. The region of nodes flooding through the network needs to be confined within logical constraints to produce better accuracy to reduce the localization error while reducing the cost. This research proposes and develops an enhanced algorithm that can improve the region area for sensor nodes placement within the specific localization area. This research aims to produce a longer network lifetime by reducing the energy used of the sensor nodes and at the same time minimize the localization error. This research describes the flooding control of the node's placement for the localization of the DV-Hop algorithm. The placement of nodes is limited inside the region of the anchor nodes. The simulation environment setup comes with parameters such as the radius, area, number of nodes, types of nodes, the anchor proportion, and the value of the regions. The result shows an improvement for the localization error, the power consumption, and the accuracy of the nodes positioning.

INDEX TERMS DV-Hop, localization, range-free localization, WSN.

I. INTRODUCTION

Wireless Sensor Networks (WSN) has become a significant technology attracting enormous research interest [1]. WSN is composed of multiple sensor nodes that monitor the physical real-world entities by having communication without devices i.e. wireless known as sensor nodes. The sensor nodes are deployed in a designated monitoring field and form a multi-hop self-configured network through wireless communication. Sensor networks represent a significant improvement over traditional sensors, which are deployed either far away from the actual phenomenon or several sensors that perform only sensing can be deployed. Sensor nodes are fitted with an on-board processor. Instead of sending the raw data to the nodes responsible for fusion purposes, sensor nodes use the processing abilities to locally perform simple computations and transmit only the required and partially processed data.

The associate editor coordinating the review of this manuscript and approving it for publication was Noor Zaman.

Ad-hoc networks have mostly been studied in the context of high mobility, high power nodes, and moderate network sizes. An Ad-hoc network can be defined as a network which deals with specific nodes under specific purposes. The Ad-hoc Positioning System (APS) done by [2] is appropriate for indoor location-aware applications, when the network's main feature is not the unpredictable, highly mobile topology, but rather deployment that is temporary, and ad-hoc. The Global Positioning System (GPS), which is a public service, can satisfy some of the above requirements. However, attaching a GPS receiver to each node is not always the preferred solution for several reasons such as cost, limited power, inaccessibility, imprecision, and the sensor size which is currently the size of a small coin.

The nodes localization algorithms can be classified into three categories that are range-based and range-free which are used to measure the actual distances between nodes. The hybrid localization is the combination of a range-free and range-based method. Centroid Algorithm (CA) and the Distance Vector-Hop (DV-Hop) algorithm uses the estimated distance instead of the metrical distance to localize the unknown nodes. For the range-free category, the localization has a considerable focus on the DV-Hop algorithm. Due to the high cost of hardware facilities and energy consumption required by range-based approaches, the range-free algorithms attract more researcher attention. Reference [3] have listed 5 protocols that provide both energy efficiency and delay resistance. Among the researches that have been done on energy efficiency are presented by [4]–[13]. Reference [14] give a comprehensive review of the cluster based on the energy-efficient routing protocols for WSN. The sensing area is divided into groups to balance the energy level of sensor nodes inside the clustering process.

The DV-Hop localization algorithm is one of the typical representatives of a range-free localization algorithm [2]. The DV-Hop algorithm has proven the ability to reduce the localization error. The enhancement of DV-Hop is centric on several elements. It is analytically proven that better accuracy can be achieved if the anchors are uniformly spread on the parameter of the network. The basic idea of DV-Hop is that the distance between the unknown nodes and the reference nodes is expressed by the product of average hop distance and the hop count. At first, several anchor nodes are properly distributed. The average hop distance is calculated in a method in which not only the global property of average hop size and anchor nodes are considered.

Most distributed localization algorithms demand less communication overhead than centralized algorithms which require relaying the connectivity or range measurements from every sensor to the base station. On the other hand, DV-Hop is the only distributed localization algorithm that has a large communication overhead because of the flooding algorithm which occurs in two phases. Thus, generating redundant communication and power consumption. Furthermore, in DV-Hop as the network size increases, the number of deployed anchors necessary for accurate localization increases. This increase in the number of anchors together with the increased communication makes DV-Hop not suitable for sensor networks used for the event monitoring and other large-scale WSNs.

This research proposes and develops an enhanced algorithm that can improve the region area for sensor nodes placement within the specific localization area to address the network flooding associated problems, whilst increasing the network lifetime and reduce power consumption while minimizing the localization error. This paper is organized into seven sections including this introductory section. Section 2 presents a detailed related work and analysis of localization algorithms which have been developed for range-free localization in WSN in particular hop based. The details of the algorithms, their respective advantage, and disadvantages are presented in Section 3. Section 4 discusses the proposed method. Section 5 presents the performance analysis technique which is the discrete-event simulation employed for this research. Section 6 presents the results and discussion and Section 7 concludes the paper with a summary and suggested future work.

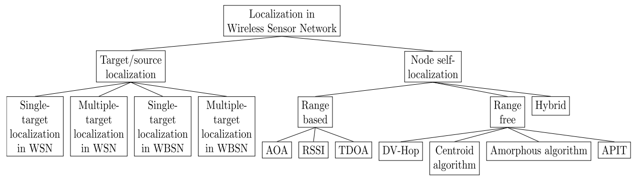

Localization in Wireless Sensor Network
Target/source localization
Node self-localization
Single-target localization in WSN
Multiple-target localization in WSN
Single-target localization in WBSN
Multiple-target localization in WBSN
Range based
Range free
Hybrid
AOA
RSSI
TDOA
DV-Hop
Centroid algorithm
Amorphous algorithm
APIT

FIGURE 1. **Localization method taxonomy.**

II. RELATED WORK

The localization in WSN can be classified into two categories that are the target/source localization and node self-localization [4]. The taxonomy of the localization methods is presented using a chart, illustrated in Figure 1. The target/source localization can be further classified into four categories which are the single-target localization, multiple-target localization, single-target localization in Wireless Binary Sensor Network (WBSN), and the multiple-target localization in WBSN. As for node self-localization, it is further classified into three categories which is the range-based localization, range-free localization which is used to measure the actual distances between nodes, and hybrid localization is the combination of the range-free and range-based method. The former method as in [2], [15] and [16] uses the measured distance/angle to estimate the location and the latter method as in [5]–[9] and [17]–[24] uses the connectivity or pattern matching method to estimate the location. This research provides the related works that have been done by researchers under the scope of the localization algorithm which is focused more on the range-free algorithm.

A. RANGE-FREE DV-HOP

Reference [2] presented the APS, a method used to extend the capabilities of GPS to non-GPS enabled nodes in a hop by hop fashion placed within an ad-hoc network. One of the aims of the positioning system is to enhance position accuracy as the fraction of landmarks of the entire population increases. APS used the landmarks to communicate between the other landmarks by using the direct signal strength measurement. The landmark has estimated a number (i.e. ≥ 3) and this is where the DV-Hop is employed which is during the second hop neighbors infer the distance to the landmarks and making the rest of the networks follows the flood in a controlled flood manner, that initiated at the landmarks.

Reference [25] proposed the Concentric Anchor Beacon (CAB) localization algorithm for wireless sensor networks. CAB is a range-free approach and uses a small number of anchor nodes which the neighboring anchors' sensors do not require to exchange their information between anchors. CAB has a low computational overhead that made it simple to implement. Each anchor transmits anchors at different power levels, where from the information received by each anchor heard, nodes can determine in which annular ring they are located within each anchor. The estimated position of the node is taken as the average of all the valid intersection points.

Aiming at the node localization positioning problem of WSN, an improved DV-Hop positioning algorithm was proposed in [26]. A general model for 2-Dimensional (2-D) position location estimation of a source using more numbers of anchor nodes was developed in this research to denote the distance between the unknown node and the anchor node. The results from this research concluded that the more regularly anchors are placed, the lower the localization average error and higher location coverage.

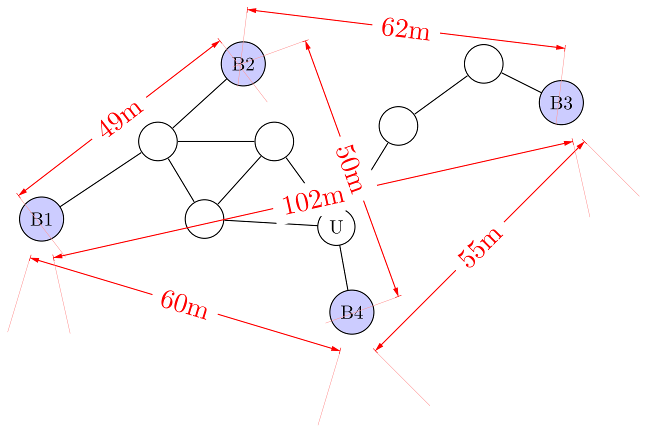


FIGURE 2. Example of hop size for Modified Dv-Hop.

Reference [27] use the original average hop size estimate scheme for a particular pair of anchor nodes. The research comes forward with a term of global part and local part to differentiate the calculation of the average hop-size in the modified DV-Hop from the [2] where the average hop size only exposes the property of hop-size for the whole network. The global part measures the global property of hop-size for the whole network, while the local part measures the local property of hop-size for a particular pair of anchor nodes. The example of global property and local property calculation is illustrated in Figure 2. Nodes B1, B2, B3, and B4 are the anchor nodes while node U is the unknown node. Node B4 is believed to be the closest (in hops) anchor node to node U. The global part of hop size estimated by B4 is calculated as $(49 + 50 + 55)/(4 + 3 + 4) = 14$. The local part of hop size B1 would be $(49/4) = 12.25$, B2 would be $(50/3) = 16.67$ and B3 would be $(55/4) = 13.75$ respectively. From this node selection, the effective hop size from B4 to B1, B2, and B3 can be calculated depends on the weighted coefficient value and the results produced the estimated distance to the three anchor nodes after the node U gets the hop sizes from B4. The results values are then plugged into the triangulation procedure for node U to get an estimated position. The research showed that the modified DV-Hop algorithm outperforms [2] algorithm, especially when the contribution of the local hop-size is selected reasonably between a particular pair of anchor nodes.

Reference [28] proposed a new localization algorithm known as Hybrid Distance Vector Hop (HDV-Hop). In the proposed algorithm, anchors are placed in a circle or a semi-circle around the perimeter of the WSN. The numbers of anchors have been minimized to the exact number where it is only deployed on the perimeter of the network which is already being determined manually. The base station calculates the coordinates of the unknown sensor as opposed to the DV-Hop where the unknown sensor itself has to calculate their coordinates. Anchors are required to unicast their calculated average hop-length to the base station and cast to their neighbors with minimum hop counter (global-minimum neighbor). Reduction in the flooded information has resulted in the reduction of redundant communication and ultimately will reduce the power consumption per sensor.

Reference [29] proposed two range-free localization approaches for static WSNs, the Mobile Beacon-assisted Localization (MBL), and the Adapting Mobile Beacon-assisted Localization (A-MBL). In MBL, the initial set of samples for each unknown node is chosen randomly from the whole deployed area and represented by a set of uniformly samples with equal weights. The weight equal to one represents the importance of the corresponding sample. The anchor's current initial position is also chosen randomly from the while deployment area. The anchor randomly moves a distance in any direction from the previous location. A new sample is generated from each current sample by randomly choosing a point within a circle centered at the current location of the sample and the radius of the real location of the unknown node. MBL is proved to be outperformed two of the previous works in static WSNs which is the Mobile and Static sensor network Localization (MSL) and Arrival and Departure Overlap (ADO) when both of them use only a single mobile anchor for localization. A-MBL is then proposed out to increase the efficiency and accuracy of MBL by adapting the size of sample sets and the parameter of the dynamic model during the estimation process. A-MBL figures out that keeping more samples can improve efficiency at the beginning of the localization.

Focusing on the Trilateration of WSNs, to overcome the effect of localization error caused by the selection of anchor nodes, a selective strategy of anchor nodes based on the angle information has been proposed by [30]. The comparative analysis of angle was used and the topology shape of the selected anchor nodes and the position relationship with unknown node was determined. In the selection process of anchor nodes, [30] firstly determined whether the subdividing angles are less than or equal to the included angles of the three anchor nodes respectively. Then, the combination of anchor nodes which is closest to the regular triangle from these anchor triangles is selected and the Trilateration positioning finally is performed. Therefore, the unknown nodes could select three anchor nodes which are the most accurate to execute the Trilateration. The basic idea of anchor nodes selection is to judge whether an unknown node lies inside an anchor triangle using the comparison of angle.

Reference [31] came up with three improved methods to resolve the location error of the classic DV -Hop localization algorithm. First, the average hop-size of each anchor node is calculated by adopting the Least Squares Method (LSM). Second, the distance between unknown nodes and anchor nodes is refined using the average hop-size of different anchor nodes. Third, the initial coordinates of unknown nodes are estimated through repeated applications of the Multilateration method which is recorded in the second step of DV-Hop, then the average of them. The performances were compared to acquire the positioning of the algorithm. It was observed from analysis and simulation experiments that the improvements achieve significantly higher performance than the classic DV-Hop algorithm at the cost of increasing appropriate computation overhead.

Reference [32] proposed an improved DV-Hop algorithm which was based on the meshless Weighted Least Square Method (WLSM). The basis of this characteristic combined with the related mathematics principle, the weighting LSM was introduced to the DV-Hop algorithm by established a weighting programmed. The basic idea of WLSM is to use the moving LSM to construct the approximate function where the penalty function method was used to impose boundary conditions and the following substituted into the governing equations. The WLSM is presented based on the Jiaoyou Meshless Better Weighted Least-Squares Method (MBWLSM) which is used to improve the method stability by the use of its matrix calculation. The method of averaging the localization results was the error produced by the network conflicting has reduced. The simulation results show that the proposed improvement enhances the localization accuracy of the nodes successfully in comparison to the other LSM.

Reference [10] made enhancements by proposing a feasible method of failure localization which can be applied on power grid cables came out with the improvements to make the hop size a constant value and were believed to be an ideal grid monitoring program. The sensor nodes lifetime is limited as the proposed method used batteries but the batteries can be charged at any time by the electromagnetic induction that can produce long-life energy.

In [16], three algorithms have improved the original DV-Hop's [2] accuracy of localization. The first improvement is based on the secondary reference node that has one hop to the anchor nodes. The second improvement focused on the node of the last hop. The third algorithm is based on the Max-Min method that was introduced instead of the multilateral measurement method. The localization error reflects the accuracy of localization.

Reference [18] developed a modified Weighted Centroid location Algorithm (WCA), which can reduce the effects of the density of anchor nodes and uniformity of deployment. This algorithm is based on the CA where the contribution of every anchor nodes is quantified by the RSSI thus makes it one of the WCA benefits. Using the weighted average of the received anchor nodes' coordinates as the estimation results allows the receivers to obtain an optimum localization result. The algorithms depend on the number of anchor nodes based on the performance of the estimation error. The positioning error decreased as the number of anchor nodes placed increased.

Reference [9] proposed Four Corners DV-Hop (FCsDV-Hop) location algorithm which utilized the anchor placement by dividing the whole area into smaller regions. The position of the unknown nodes is calculated by using four anchor nodes which are the closest to it. The placement of four anchor nodes at probable four corners of the small region is considered by the factors that exist in the physical environment. The simulation environment models are made up of all sensor nodes that have a perfect spherical radio propagation and identical transmission range. For the ideal model, all of the sensor nodes are at their fixed position while for the random model, all of the sensor nodes are randomly placed.

In the research by [17], each node emits a multi-power anchor within its super-frame time to reduce localization error. The super-frame time works as the anchor nodes transmitting multi-power anchor signals emits signals sequentially within an anchor signal period throughout the time. The signals will be active and idle where the idle situation is double the active time which follows by another anchor signal period after that. Reference [17] designed a new hop to increase the value by the ratio of the range according to the power of the anchor, rather than the usual increment of 1. The Radio Irregularity Model (RIM) was adapted and the robustness showed proved the adequate environment.

Reference [33] did a comprehensive review of 31 WSN simulators. Among the simulators are in [6] and [34]. Reference [6] conducted a large number of experiments using the WSN simulator known as Network Topology (NetTopo) by implementing the Hop-Count-Ratio-based Localization (HCRL) algorithm by [35] with Connected K-Neighbourhood (CKN). The radio range irregularity model was used to reflect the impact of the duty cycle and this models' impact on the DV-based localization algorithms. The HCRL algorithm used only the ratios of anchor-to-node Hop Counts (HC) to do localization and has been able to satisfy the low cost. The CKN algorithm was utilized to ensure the sleep scheduling of the network and at the same time also consider the effect of the node's Degree of Irregularity (DOI). The HCRL algorithm which is based on the CKN algorithm acknowledged that sleep time is required before or after transmitting an FM to save energy.

Reference [34] introduced a node localization algorithm based on a moving anchor node. Utilizing the random generation function rand of Matlab, 20 random distributions were produced on behalf of the unknown nodes and a period T ranging from 1 to 10 units of time. The simulation results showed that the anchor node movement trajectory and mobile anchor node sending of location information grouping cycle are key factors in influencing the node location accuracy and respective communication cost.

The Localization algorithm for Large Scale in Wireless Sensor Network (LLSiWSN) proposed by [5] is an algorithm with a target of achieving minimum or no anchor nodes case. The simulation was set with 210m $\times$ 210m areas and the sensor nodes are spread into the area randomly and equally, assumed to be static. The numbers of nodes are taking into account as too large and the pre-locating to all the nodes divided into groups of an equal number of nodes, which is the group size to the prevailing circumstances.

Reference [36] proposed the Improved Weighted Connectivity (IWC) based algorithm as an effort to improve the fundamental CA as already briefly discussed in [18]. There are two types of the proposed IWC CA, which are the Connectivity-Based Centroid (CBC) and Improved Connectivity-based Centroid (ICB) respectively. In CBC, the difference of distance between the unknown-reference node and the communication range of the unknown will be utilized by the unknown node itself. The ICB Centroid divides the positions of reference nodes inside the communication range of unknown into two parts. The first one is those nodes in the coverage area which is less than half of the communication range. The second one is the nodes which range between half of the communication range up to the maximum communication range.

The range-free system needs less equipment, has less cost and energy consumption, so the range-free solution is the main direction in large-scale sensor network localization. Reference [7] proposed a range-free system localization method which is like proposed by [5] that focused on the large-scale network. To solve the problems of distance estimation, the anchor border network was located further from the other anchors as the anchor placement strategy. A two-step Weighted DV-Hop (WDV-Hop) algorithm applied which is the message-by-step shortest path calculation and each regular node calculates its location only once. This is because the anchor calculates the heat source to overcome the energy and storage space problems.

An improved Shortest Distances DV-Hop (SDDV-Hop) algorithm is proposed in [20] by modifying the network average hop distance to improve the location accuracy. The shortest-path distance is selected as opposed to the smallest hop count path and views the ratio between shortest-path distance and the straight-line distance as the contribution degree of straight-line distance. This is used to modify the entire network average hop size calculation.

Reference [37] proposed an algorithm was to establish the simultaneous equations with the ratio of distance and path length. The collinearity of the groups of anchor nodes is calculated before positioning and the non-collinear groups of nodes are used to locate the unknown nodes. The collinearity of anchor nodes is concerned with the localization error i.e., the growth of the area according to [11]. When the anchor nodes are chosen, the collinearity of the three anchors is calculated and the anchor nodes groups are used to locate the unknown node. The energy consumption of broadcasting the one average size of one hop is reduced but it increases the complexity of the algorithm. The improved algorithm saves substantial energy and it reduces the signal conflict.

Reference [38] placed known nodes by Manual Distance Vector-Hop placement (MDV-Hop) in various locations as a measure to acquire the best result. When an abnormal event occurs, the sensor node detecting the event needs the position information to locate the abnormal event and report to the base station. Without position information, the WSN is unable to function properly. By using the manual displacement of the anchor sensors in specific locations, the accuracy of the system was improved and the error of nodes in epoch performance was able to be decreased.

Reference [39] improved the DV-Hop algorithm by using three approaches. First, the distribution of nodes for some anchor nodes was set or fixed at the borderland of monitoring regions, whilst the rest were distributed randomly. Second, the average one-hop distance used by each unknown node for estimating its location is modified through weighting the received average one-hop distances from anchor nodes. The HopSize formula used is as follows. Third, the computation of the final location uses the Particle Swarm Optimization (PSO) to correct the position estimated by the 2-D Hyperbolic localization algorithm, which makes the result closer to the actual position.

There are essentially two main categories of sensor nodes network topology, even and random. Even topologies distributed sensor nodes and anchors over the deployment area in an exact grid, whilst random topologies trouble individual nodes positions on the grid with random noise. Random topologies however have a better reflection on the deployment scenarios in real-world environments. These categories of topologies then can be classified into two types, the regular shapes and the irregular shapes which are according to the regularity of the placement densities and shapes [40]. Among these are the C-shaped, L-shaped, and ring-shaped topologies which are typical irregular topology examples. C shape topology resulted in the lowest error and this is good because the performances performed by C shape are better than other topologies, thus give better accuracy of the evaluated method.

Reference [41] proposed a new distributed range-free Localization Algorithm based on Self-Organizing Maps (LS-SOM). Utilizing the intersection areas between radio coverage of neighboring nodes has been the strategy of the algorithm to maximize the correlation between neighboring nodes in the distributed implementation of Self-Organizing Maps (SOM). The proposed method increased the quality of the topology estimation based on the information from the heard anchors and reduced the time of the topological convergence. The anchors that participate in localization will be flooded inside the network just one time so that the nodes can calculate the initial location for fast topology convergence.

Reference [42] proposed an improved DV-Hop location algorithm based on the reduced accumulation errors of the average hop distance of the unknown nodes in WSNs using two strategies. First, the anchor node broadcast their location information packet and initialized it to zero. The receiving node records the minimum number of hops for each anchor nodes and ignores the packet with a bigger number of hops from the same anchor. Then the hop value plus 1 and forwarding the packet to the neighbor nodes. With this strategy, all nodes record the minimum hop number for each anchor node. Second, after having the minimum hop number of all the anchor nodes, the unknown node will be set the weight to balance the cumulative error of each anchor node. Then the value of the hops broadcast for the whole network. To improve the positioning accuracy of the unknown node, the centroid of the estimated region was set as the coordinates of the unknown node.

A WDV-HOP algorithm is proposed by [43] that weighted forward on basis of the DV-HOP Algorithm to confirm the changes of localization accuracy. All average hop size values of the anchor node use the distance between the anchor nodes to make a weighted calculation and achieve the final value which was assumed to be as farther the distance, the greater the weights. The method gave more accurate localization value in the network. The number of deployed anchor nodes and the transmitting power of the node served as the controlled parameters with Trilateration as the distance calculation in the localization algorithm. The LSM was adopted to help to solve the calculation on finding the distance.

Reference [44] algorithm employs the anchors to adaptively search for an optimal packet reception ratio (PRR) threshold to adjust the RND-based distance estimation. Reference [45] proposed the RAPS algorithm, where reliable anchor pairs are selected for localization by observing their average hop progresses. The distances to the reliable anchors are estimated based on the geometric approximation of the node location. RAPS algorithm proved to give accuracy with fewer anchors while having less communication overhead in anisotropic networks.

Reference [46] proposed a stochastic log-normal RSS model to describe the fading effects caused by the object/human movement in both indoor and outdoor environments. Both the average path-loss and the fluctuations of the received signal strength induced by the moving target are jointly modeled based on the theory of diffraction. The model is proved to be tight enough to be adopted for real-time estimation of the target location.

Reference [47] design LiFS, an indoor localization system work on infrastructure and mobile phones. A system of fingerprint space was created in which fingerprints are distributed according to their mutual distances in the real world. The fingerprint space is then automatically mapped to the floor plan in a stress-free form, which results in fingerprints labeled with physical locations thus achieved location accuracy. Reference [48] then proposed a system to extract the information about human location, motion, and health-critical posture features and this help in term of workers safety and protection in industrial application.

Reference [49] use historical beacons to came out with a localization algorithm that required only a one-hop beacon to broadcast. Reference [50] and [51] came out with an anisotropic network localization idea. Reference [50] propose an algorithm that is robust against the anisotropic signal attenuation present in any wireless channel, and develop a new distance estimation (DE) approachable to efficiently derive distances' estimates in closed form. Reference [51] then introduced a control parameter call MaxHop in the first phase of the DV-Hop algorithm and resulted in better localization accuracy for anisotropic networks, with much faster convergence and low overheads.

Reference [52] develop an algorithm by deriving the average location estimation error (LEE) in closed-form and able to achieve the LEE average and variance of about 0 when the number of sensors is large. Reference [53] use a technique to determine the number of virtual nodes together with their positions in grid form and uniformly random positioning to get the most precise location. Reference [54] proposed a localization algorithm that uses a grid scan approach to avoid the complex geometric computations in an initial residence area. The centroid of the refined residence area is used to estimate the node location in this research.

Underwater localization becomes a trend in 2019 as proposed in [55]– [57] with consideration of the realistic case. In [55], the sensor nodes transmit their data through beacons and then through the surface buoys antenna. Reference [56] proposed a received signal strength (RSS)-based localization framework for energy harvesting underwater optical WSN (UOWSNs). The sensor nodes with insufficient battery harvest ambient energy and start communicating once they have sufficient storage of energy. Reference [57] model UOWSNs as uni-directional random sector graphs where the connection between sensors is established by point-to-point directed links with the coverage region of each node is angular-sector shaped.

In the era of the Internet of Things (IoT), [58] came with a lightweight lattice signcryption method to reduce the cost and complexity but with desirable accuracy. In the meanwhile, [59] enhance indoor localization by using the multimodal sensing via two images, the internal measurement units (IMU) sensors reading and WiFi signal to locate the user point of interest in two pictures taken at the same location.

B. HYBRID: RANGE ASSOCIATED HOP BASED LOCALIZATION

Reference [8] proposed a mobile anchor algorithm that has the deployment area which was partitioned into hexagon cells to achieve the shortest path. The shortest path is important to make sure that the anchor nodes cover the entire area within the shortest time and consume the minimum energy. An optimal movement schedule for the anchor node was developed, which achieved the shortest path under expected localization accuracy using cellular hexagon cells.

Given the error problem resulting from the approximate calculation when there is only a hop distance between an anchor node and the location node in the DV-Hop algorithm of the wireless sensor network, [60] introduced the RSSI ranging technology which was derived based on the original algorithm. The RSSI-ranging technology will give information on whether the anchor node is only a hop away from the location node so that the use of the DV-Hop algorithm method can be decided after that. The anchor nodes are directly assumed to be a hop away between each other and any node that receives the RSSI packets that broadcasted in the network can compute the RSSI distance from itself to the anchor.

Reference [24] presents a distributed algorithm for node localization based on the Newton method for WSNs. Utilizing the basis of the pairwise distance measurements and the local information received from the neighboring nodes, each node updates its location estimate by finding a slope step using the Newton method rather than using only the local information and then broadcasts the updated estimate to all the neighboring nodes. Once the location estimate is updated, the sensor node broadcasts the updated estimate to all the neighboring nodes and terminates when the update step which is computed less than a predefined threshold.

Reference [61] proposed a localization algorithm based on Trilateration called Localization Mobile Anchor Trilateration (LMAT). In the LMAT, the anchor node work as mobile as a need to traverse the entire WSN along the boundaries of the equilateral triangles. The proposed equilateral triangle trajectory can guarantee that all the unknown nodes receive their message packets and obtain their estimated positions. In comparison to the random movement of the mobile anchor node, the LMAT has significantly reduced localization error.

Reference [19] proposed a localization algorithm for wireless sensor networks that use only one mobile station to find 100 unknown nodes position with high accuracy and acceptable speed. The combination of radio frequency and supersonic is used to measure the distance instead of using a regular method like other proposed algorithms [62]–[64] which used radio frequency and ultrasonic signals. The high performance of the proposed algorithm is clear because only three distances are needed to the base points as opposed to other algorithms that require at least four base points. Therefore, the proposed algorithm was less affected by the internal transfer of the package with lower throughput because the power of fixed nodes was reduced. These ideas have resulted in the neighbors being able to estimate their position by receiving messages and calculating receive signal strength indicator.

The GPS-Free-Free (GFF) algorithm proposed by [15] applied the idea similar to the distance-vector routing in the DV-Hop algorithm to estimate the distance between nodes through the number of hops. Based on the algorithm proposed by [2] and [15] research ideology on the measurement of distance, [23] proposed a node position control algorithm based on RSSI technology. This algorithm used target parameter measurement quantification and relating geometric knowledge to compute the distance. The experimental results showed that this method can complete a complex computation process with fewer parametric variables.

Reference [12] discovered that the consistent usage pattern of Local Aggregators (LAs) attains a pre-specified lowest possible energy level which leads to a phenomenon "HOT SPOT" in their research. In a query-based protocol, every query is passed through the initially selected LAs. A scheme was proposed to overcome the hot spot effect with low overhead. A node with high communication overhead will consume more energy. Every query is passed through the initially selected LAs in the query-based protocol. It was concluded that properly framed heuristics which utilizes residual energy level of sensor nodes and binary location index may infer avoidance of the hot spot effect till the network attains its targeted span of a lifetime.

Reference [21] proposed a framework to implement a dynamic path of localization in WSN using the efficiency and effectiveness of the anchor nodes. The localization was done by using the distance to calculate the weight for the next position. The location of the next neighbor was decided by the previous nodes' location. Only one neighbor of the current Mobile Beacon Position (MBP), which is the anchor node position, is selected as the target and the other neighbor is considered as a spread inside the area.

Reference [22] proposed another algorithm for localization of WSN based on the machine nature of a anchor nodes. Reinforcement Learning (RL) is adapted to function as the brain of anchor nodes driving the machine to localize the unknown sensors.

In [11] an unsupervised and auto-acting anchor nodes algorithm for localization of WSN was proposed. Compared with the previous algorithm in [21], the methods change the implementation of a dynamic path comprehensively. The proposed method employed direction by area and cosine similarity instead of using distance on calculating the weight for the next position. This method used the distribution and also the relationship of the settled sensors to assist anchor nodes in choosing the next position. Based on the framework proposed in [11], [22] reengineered the algorithm to find the best position for an anchor node by grouping the weight of cosine to reflect the relation between related sensors. The method was more lightweight as it avoided any complicated computation such as filters and graph theory as in [22].

III. CONSTRAINTS OF THE DV-HOP

Most range-free localization algorithms assume that only a small fraction of sensors have their absolute positions either manually configured or through GPS receivers. Among which are the localization algorithms proposed by [4], [9], [13], [65], [66].

Reference [9] state that the beacon nodes in the DV-Hop algorithm need to broadcast beacon messages to the entire wireless sensor networks, and the nodes which receive the beacon message is also obligated to broadcast the distance. Then they need to broadcast the message again once the beacon nodes get the average one-hop distance. The broadcasting, therefore, is done twice which causes lots of communication, and these waster lots of the node's energy. Therefore, a good localization algorithm would optimize the amount of communication.

A localization technique that uses hop-count information can minimize the number of anchor nodes because the position of the anchor node can be propagated by packet forwarding [13]. However, it is inefficient to be applied to the real sensor network system in terms of overhead, accuracy, and simplicity. The HCRL algorithm by [13] presents an enhanced localization algorithm performance while consuming less energy in comparison to [9]. The experiments also demonstrated that the localization performance can be improved by the hop-subdivision through transmission power level control.

The Range-free system needs less equipment, has less cost and energy consumption, so a range-free solution is the main direction in large-scale sensor network localization. At the same time, the randomly placed anchor nodes also make some of the formal parts very close to even a lot of foundation. Thus, this will lead to badly estimating the distance of these anchors, because they are too close to the target and the hop distance becomes impossible to be counted [4]. The broadcast and anchor hop size calculations occur more than once, and this is not effective for the node energy and storage. The broadcasting of each node is maintained by receiving the first value of the information periodically and the subsequent message is ignored to avoid the endless flood. The high performance of anchor nodes is needed and anchor nodes depend on the distance measurement errors. From [2], it is clearly stated that the accuracy of the algorithm is affected by the distances between the unknown nodes and the anchor nodes. If the nodes are not uniformly distributed or there is a position where no nodes existed, the accuracy of position information becomes poor [66]. The generated packets of nodes information need to be equally flooded in the network to limit the region of the nodes when computing its position so that the positioning of the nodes becomes more accurate and reduces the localization error [65]. Table 1 summarizes the constraints of the previous enhanced ideas of positioning algorithms.

TABLE 1. Constraints summary of previous enhanced ideas of positioning algorithms.

Parameters \ References	[1]	[4]	[5]	[6]	[7]
Nodes $\geq$ 100	✓	✓	✓	×	✓
Radius stated	×	×	✓	×	✓
Mobility model	✓	✓	×	×	×
Sensor node topology	✓	✓	✓	✓	✓
Node position corrections	✓	✓	✓	✓	✓
Energy expenditure	×	×	×	✓	✓
Flooding	✓	✓	✓	✓	✓

Table 1 shows a summary of the constraints of the previous enhanced idea of DV-Hop. The analysis of Table 1 shows us that the existing positioning algorithms used to correct the position of the nodes inside their topology to control the nodes flood inside the network. The mobility aspects are considered as present when the number of nodes is flooded within the area of the node's radius. However, the energy used was not explained for the mobile nodes.

IV. PROPOSED FCDV-HOP

FCDV-Hop has been proposed as an algorithm that has high positioning accuracy by controlling the network flood movement of the sensor nodes and the placement of the neighbor is identified for each of the anchors. Each of the anchors are connected as the backbone of the network flood movement. The energy consumption for the nodes and the localization is low compared to the previous work. The DV-Hop fundamental proposed by [2], the phase starts with the distribution of the nodes and the anchor. Three beacon nodes are then used to determine the position of the unknown nodes by using hops. While the beacon nodes flood to the anchor inside the network, the information about the sensor is transferred. In this research, after the determination of the anchor nodes position, the other nodes will be located. The placement of the beacon nodes for nodes with an angle will be performed based on the fixed position on the anchor nodes region. The other beacon nodes are also distributed inside the anchor nodes region. The beacon node is distributed inside the anchor region to limiting the nodes involved in the flood during the broadcasting. The information of the nodes' location is transferred to the anchor nodes while broadcast.

Algorithm 1 presents the pseudo-code of the proposed FCDV-HOP. The parameter for *A: Anchor, N: Node, AH: Area Height, AW: Area Width, D: Dimension, C: Cloud,* γ*N: Angle Node,* and δ*: Distance* is initialized at the start.

The initialization begins by setting the parameter D for the area by the parameter AH_2 times the parameter AW_2. Then, the parameter A value was assigned by calculating the value of the parameter TN times the parameter Ap. The parameter N *is* valued from the parameter TN minus parameter A. The parameter C produces its value by dividing the parameter N by parameter A. The clock then begins to start by using the method of startTime() to calculate the time of the simulation. The area height will have the start and the end value where the start value assigned as 0 and the end value is based on the field area height. This concept is also applied to the area width. Coordinate generated for the anchor in the simulation are random based on the end value for the area height and the end value for the area width. After having all of the coordinate for the anchor, the anchorZone() the method will be apply shown by Algorithm 3. Each anchor zone then will be assumed as a cloud.

For the FCDV-Hop algorithm, there are 2 types of nodes are used to be tested which are the nodes and the angle nodes. Inside the cloud, the nodes will be randomly generated based on the anchor zone. A normal node is defined as the node which is generated inside the anchor zone but without the angles, while the anchor nodes are defined as the nodes

Algorithm 1 FCDV-Hop Algorithm

(TN: Total Node, Ap: Anchor Proportion)

```
1:  Initialize parameters
2:  $D = AH_2 * AW_2$
3:  A = TN * Ap
4:  N = TN - A
5:  C = N / A
6:  Anchor generated
7:  Nodes generated
8:  Apply anchorZone()
9:  Nodes with angle generated
10: for each object $j \in \mathcal{A}$ do
11:     for each object $i \in \mathcal{C}$ do
12:         $X_N \leftarrow random(axisX1_j \text{ to axis } X2_j)$
13:         $Y_N \leftarrow random(axisY1_j \text{ to axis } Y2_j)$
14: for each object$i \in ɣN$, where $(1 \le i \le 4)$ do
15:     $X1_{ɣN} \leftarrow random(X_{A_i} \text{ to } axisX2_i)$
16:     $Y1_{ɣN} \leftarrow Y_{A_i}$
17:     $X2_{ɣN} \leftarrow random(axisX1_i \text{ to } X_{A_i})$
18:     $Y2_{ɣN} \leftarrow Y_{A_i}$
19:     $X3_{ɣN} \leftarrow X_{A_i}$
20:     $Y3_{ɣN} \leftarrow random(Y_{A_i} \text{ to } axisY2_i)$
21:     $X4_{ɣN} \leftarrow X_{A_i}$
22:     $Y4_{ɣN} \leftarrow random(axisY1_i \text{ to } Y_{A_i})$
23: for each object $i \in N$ do
24:     for each object $j \in A$ do
25:         $\delta N_{ij} = \sqrt{(X_{N_i} - X_{A_j})^2 + (Y_{N_i} - Y_{A_j})^2}$
26: for each object $i \in ɣN$ do
27:     for each object $j \in A$, where $(1 \le k \le 4)$ do
28:         $\delta_k ɣ N_{ij} = \sqrt{(X_{k_{ɣN_i}} - X_{k_{A_j}})^2 + (Y_{k_{ɣN_i}} - Y_{k_{A_j}})^2}$
29: Apply findMin() for nodes and angle node
30: Apply connectTo()
31: Apply matrixMultiply()
```

generated inside the anchor zone but with fixed angles. The anchor height and the anchor width will have the start and the end value where this value will mark the anchor zone.

An angle node consists of 4 different angles 90°, 180°, 270°, and 360° respectively. Figure 3 illustrates the position of the angle nodes to the anchor node. If there are 4 anchor nodes, the number of angle nodes will be 16 as there would be 4 for each of the anchors' nodes. The 90° and 270° angles will take place at the anchors' horizontal line while the 180° and 360° at the anchors' vertical line. For the 90° angle nodes, the x value is randomly generated at the x-axis between the anchor nodes to the end value of the anchor width area and the y value is the same as the anchor node. For the 270° angle nodes, the x value is randomly generated at the x-axis between the start value of the anchor width area to the anchor nodes and the y value is the same as the anchor node. For the 180° angle nodes, the x value is the same as the anchor node while the y value is randomly generated at the

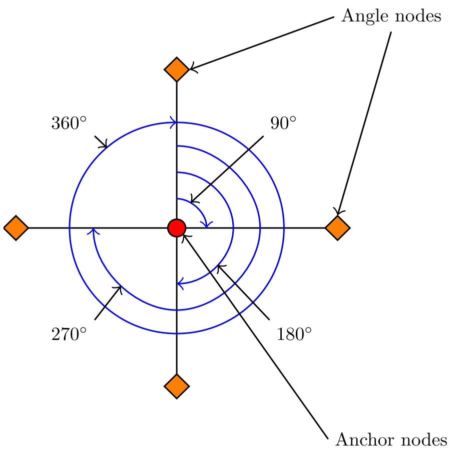


FIGURE 3. **The diagram of angle nodes positioning.**

y-axis between the anchor nodes to the end value of the anchor height area. For the 360° angle nodes, the x value is the same as the anchor node while the y value is randomly generated at the y-axis between the start values of the anchor height area to the anchor nodes.

Algorithm 2 findMin() Algorithm

```
1:  Initialize ɱ: Minimum Value
2:  $\text{ɱ}_n$ = Double.MAX_VALUE where $(1 \le n \le 3)$
3:  for each object $j \in TN$ do
4:      if j = i
5:          continue line 3
6:      if $(\delta_{ij} < \text{ɱ}_3$ and $\delta_{ij} \ge \text{ɱ}_2)$
7:          $\text{ɱ}_3 = \delta_{ij}$
8:          $\eta_3 = j$
9:      if $(\delta_{ij} < \text{ɱ}_2$ and $\delta_{ij} \ge \text{ɱ}_1)$
10:         $\text{ɱ}_3 = \text{ɱ}_2$
11:         $\eta_3 = \eta_2$
12:         $\text{ɱ}_2 = \delta_{ij}$
13:         $\eta_2 = j$
14:     if $(\delta_{ij} < \text{ɱ}_1)$
15:         $\text{ɱ}_3 = \text{ɱ}_2$
16:         $\eta_3 = \eta_2$
17:         $\text{ɱ}_2 = \text{ɱ}_1$
18:         $\eta_2 = \eta_1$
19:         $\text{ɱ}_1 = \delta_{ij}$
20:         $\eta_1 = j$
```

After locating the nodes and the angle nodes coordinate, each of the nodes will calculate their distances to the anchors by using the standard distance calculation shown by (1). The nodes calculate their distances to anchors and are followed by the angle nodes to calculate the distances to anchors. After getting all of the distances, the findMin() method was applied for both types of nodes. The findMin() method has an array that was used to find the three minimum distances for each of the nodes. The steps apply in FCDV-Hop for findMin() method is the same as the findMin() algorithm explain in Algorithm 2. The connectTo() method also applied

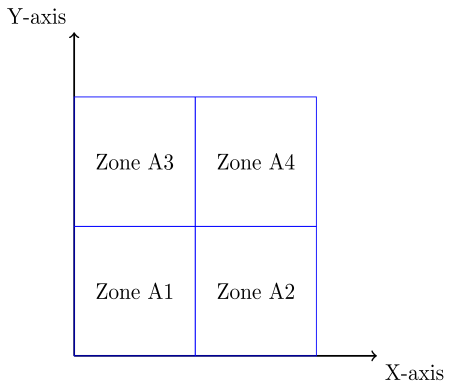


FIGURE 4. The zone classification for the anchor.

after finish the findMin() method to find out to which anchor the nodes and angle nodes connect to. The algorithm is followed by the matrixMultiply() method where it is used to find the estimated coordinate. The algorithm time end by using the endTime() method and this concludes that the simulation process is already finished. The times used throughout the calculation are calculated by using (2).

$$distance_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}, \quad (i = A, B, C) \tag{1}$$

$$Simulation\,time = endTime() - startTime() \tag{2}$$

A. CREATING THE anchorZone() METHOD FIRST STAGE

Figure 4 illustrates the zone classification for the anchor position. The anchorZone() method is divided into three stages presented by Algorithm 3, Algorithm 4, and Algorithm 5. The first stage showed by Algorithm 3 is where to determine the position of the anchor-based on the start and end values of both heights and widths. The algorithm starts with dividing the area into 2 to determine the x-axis range and y-axis range by using the calculation shown by (3) and (4). At the first stage, the algorithm clustered the position of the anchor by illustrating it in 4 different cases.

$$X_{\text{Axis Range}} = \left[\frac{\text{(End Value - Start Values) of field area width}}{2}\right] \tag{3}$$

$$Y_{\text{Axis Range}} = \left[\frac{\text{(End Value - Start Values) of field area width}}{2}\right] \tag{4}$$

1) CASE 1 OF THE anchorZone() METHOD FIRST STAGE

The first case displayed in Figure 5 where the condition of the anchor coordinate for the x-axis point is less than or equal to the x-axis range calculated earlier using (3) and the y-axis point is also lesser than or equal to the y-axis range which is calculated earlier using (4). The A1 denotes the start value of the anchor zone range width and the value is given by the start value of the area width. The C1 indicates the end value of the anchor zone range width and the value is given by the start value of the area width plus the x-axis range value. The D1 represents the start value of the anchor zone range height

Algorithm 3 anchorZone() Algorithm First Stage

```
1:  Initialize $X^R$: X-axis Range, $Y^R$: Y-axis Range
2:  $X^R = (AW_2 - AW_1)/2$
3:  $Y^R = (AH_2 - AH_1)/2$
4:  for each object $i \in A$ do
5:      if ($X_{A_i} \le X^R$ and $Y_{A_i} \le Y^R$)
6:          $A1 = AW_1$
7:          $C1 = AW_1 + ((AW_2 - AW_1)/2)$
8:          $D1 = AH_1$
9:          $F1 = AH_1 + ((AH_2 - AH_1)/2)$
10:         axisX1 $\leftarrow A1$
11:         axisX2 $\leftarrow C1$
12:         axisY1 $\leftarrow D1$
13:         axisY2 $\leftarrow F1$
14:     if ($X_{A_i} > X^R$ and $Y_{A_i} \le Y^R$)
15:         $A2 = X^R$
16:         $C2 = AW_2$
17:         $D2 = AH_1$
18:         $F2 = AH_1 + ((AH_2 - AH_1)/2)$
19:         axisX1 $\leftarrow A2$
20:         axisX2 $\leftarrow C2$
21:         axisY1 $\leftarrow D2$
22:         axisY2 $\leftarrow F2$
23:     if ($X_{A_i} \le X^R$ and $Y_{A_i} > Y^R$)
24:         $A3 = AW_1$
25:         $C3 = AW_1 + ((AW_2 - AW_1)/2)$
26:         $D3 = Y^R$
27:         $F3 = AH_2$
28:         axisX1 $\leftarrow A3$
29:         axisX2 $\leftarrow C3$
30:         axisY1 $\leftarrow D3$
31:         axisY2 $\leftarrow F3$
32:     if ($X_{A_i} > X^R$ and $Y_{A_i} > Y^R$)
33:         $A4 = X^R$
34:         $C4 = AW_2$
35:         $D4 = Y^R$
36:         $F4 = AH_2$
37:         axisX1 $\leftarrow A4$
38:         axisX2 $\leftarrow C4$
39:         axisY1 $\leftarrow D4$
40:         axisY2 $\leftarrow F4$
```

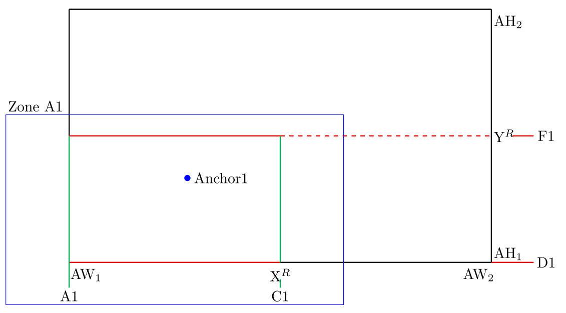


FIGURE 5. First case inside the anchorZone() method.

Algorithm 4 anchorZone() Algorithm Second Stage

```
(S: Status)
1: Initialize tS: Temporary Size; iL: List Size
2: S = 2, where 1 = different zone, 2 = same zone
3: do
4: for each object $j \in (tS - 1)$ do
5:     iL = 0;
6:     for each object $k \in tS$, where $(k = j + 1)$ do
7:         if $(axisX1_j = axisX1_k)$ & $(axisX2_j = axisX2_k)$ & $(axisY1_j = axisY1_k)$ & $(axisY2_j = axisY2_k)$
8:             if $(iL_j \not\leftarrow j)$
9:                 $iL \leftarrow j$
10:            $iL \leftarrow k$
11:    for each object $m \in iL$ do
12:        if $(zone \leftarrow iL_m) = 1$
13:            $X^R = ((axisX2 \leftarrow iL_m)$ - $(axisX1 \leftarrow iL_m))/2$
14:            $Y^R = ((axisY2 \leftarrow iL_m)$ - $(axisY1 \leftarrow iL_m))/2$
15:        if $(zone \leftarrow iL_m) = 2$
16:            $X^R = (axisX1 \leftarrow iL_m) + ((axisX2 \leftarrow iL_m)$ - $(axisX1 \leftarrow iL_m))/2$
17:            $Y^R = ((axisY2 \leftarrow iL_m)$ - $(axisY1 \leftarrow iL_m))/2$
18:        if $(zone \leftarrow iL_m) = 3$
19:            $X^R = ((axisX2 \leftarrow iL_m)$ - $(axisX1 \leftarrow iL_m))/2$
20:            $Y^R = (axisY1 \leftarrow iL_m) + ((axisY2 \leftarrow iL_m)$ - $(axisY1 \leftarrow iL_m))/2$
21:        if $(zone \leftarrow iL_m) = 4$
22:            $X^R = (axisX1 \leftarrow iL_m) + ((axisX2 \leftarrow iL_m)$ - $(axisX1 \leftarrow iL_m))/2$
23:            $Y^R = (axisY1 \leftarrow iL_m) + ((axisY2 \leftarrow iL_m)$ - $(axisY1 \leftarrow iL_m))/2$
24:        if $((X_A \leftarrow iL_m) \leq X^R)$ & $((Y_A \leftarrow iL_m) \leq Y^R)$
25:            $A1 = axisX1 \leftarrow iL_m$
26:            $C1 = (axisX1 \leftarrow iL_m) + ((axisX2 \leftarrow iL_m)$ - $(axisX1 \leftarrow iL_m)/2)$
27:            $D1 = axisY1 \leftarrow iL_m$
28:            $F1 = (axisY1 \leftarrow iL_m) + ((axisY2 \leftarrow iL_m)$ - $(axisY1 \leftarrow iL_m)/2)$
29:            $axisX1 \leftarrow iL_m, A1$
30:            $axisX2 \leftarrow iL_m, C1$
31:            $axisY1 \leftarrow iL_m, D1$
32:            $axisY2 \leftarrow iL_m, F1$
33:            $zone \leftarrow iL_m, 1$
34:        if $((X_A \leftarrow iL_m) > X^R)$ & $((Y_A \leftarrow iL_m) \leq Y^R)$
35:            $A2 = X^R$
36:            $C2 = axisX2 \leftarrow iL_m$
37:            $D2 = axisY1 \leftarrow iL_m$
38:            $F2 = (axisY1 \leftarrow iL_m) + ((axisY2 \leftarrow iL_m)$ - $(axisY1 \leftarrow iL_m)/2)$
39:            $axisX1 \leftarrow iL_m, A2$
40:            $axisX2 \leftarrow iL_m, C2$
41:            $axisY1 \leftarrow iL_m, D2$
42:            $axisY2 \leftarrow iL_m, F2$
43:            $zone \leftarrow iL_m, 2$
44:        if $((X_A \leftarrow iL_m) \leq X^R)$ & $((Y_A \leftarrow iL_m) > Y^R)$
45:            $A3 = axisX1 \leftarrow iL_m$
46:            $C3 = (axisX1 \leftarrow iL_m) + ((axisX2 \leftarrow iL_m)$ - $(axisX1 \leftarrow iL_m)/2)$
47:            $D3 = Y^R$
48:            $F3 = axisY2 \leftarrow iL_m$
49:            $axisX1 \leftarrow iL_m, A3$
50:            $axisX2 \leftarrow iL_m, C3$
51:            $axisY1 \leftarrow iL_m, D3$
52:            $axisY2 \leftarrow iL_m, F3$
53:            $zone \leftarrow iL_m, 3$
54:        if $((X_A \leftarrow iL_m) > X^R)$ & $((Y_A \leftarrow iL_m) > Y^R)$
55:            $A4 = X^R$
56:            $C4 = (axisX2 \leftarrow iL_m)$
57:            $D4 = Y^R$
58:            $F4 = axisY2 \leftarrow iL_m$
59:            $axisX1 \leftarrow iL_m, A4$
60:            $axisX2 \leftarrow iL_m, C4$
61:            $axisY1 \leftarrow iL_m, D4$
62:            $axisY2 \leftarrow iL_m, F4$
63:            $zone \leftarrow iL_m, 4$
```

Algorithm 5 anchorZone() Algorithm Third Stage

```
1: for each object $m \in (tS - 1)$ do
2:     $a = axisX1_m$
3:     $b = axisX2_m$
4:     $c = axisY1_m$
5:     $d = axisY2_m$
6:     for each object $n \in tS$, where $(n = m + 1)$ do
7:         $e = axisX1_n$
8:         $f = axisX2_n$
9:         $g = axisY1_n$
10:        $h = axisY2_n$
11:        if $(a \neq e)$ & $(b \neq f)$ & $(c \neq g)$ & $(d \neq h)$
12:            S = 1
13:        if $(a = e)$ & $(b = f)$ & $(c = g)$ & $(d = h)$
14:            S = 2 then stop line 1
```

and the value are given by the start value of the area height. Lastly, the F1 signifies the end value of the anchor zone range height, and the value is given by the start value of the area height plus the y-axis range value. After getting all of the zone value, the A1 will be the axisX1 which is the start point and the C1 will be the axisX2 which is the endpoint of the anchor zone at the x-axis. The D1 will be the axisY1 which is the start point and the F1 will be the axisY2 which is the endpoint of the anchor zone at the y-axis.

2) CASE 2 OF THE anchorZone() METHOD FIRST STAGE

The second case is displayed in Figure 6 where the condition of the anchor coordinate for the x-axis point is more than the

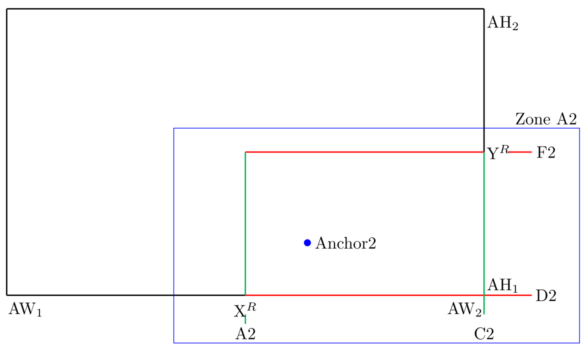


FIGURE 6. Second case inside the anchorZone() method.

x-axis range calculated earlier using (3) and the y-axis point is lesser than or equal to the y-axis range which calculated earlier using (4). The A2 denotes the start value of the anchor zone range width and the value is given by the x-axis range value. The C2 indicates the end value of the anchor zone range width and the value is given by the end value of the area width. The D2 represents the start value of the anchor zone range height and the value given by the start value of the area height. Lastly, the F2 signifies the end value of the anchor zone range height and the value given by the start value of the area height plus the y-axis range value. After getting all of the zone value, the A2 will be the axisX1 which is the start point and the C2 will be the axisX2 which is the endpoint of the anchor zone at the x-axis. The D2 will be the axisY1 which is the start point and the F2 will be the axisY2 which is the endpoint of the anchor zone at the y-axis.

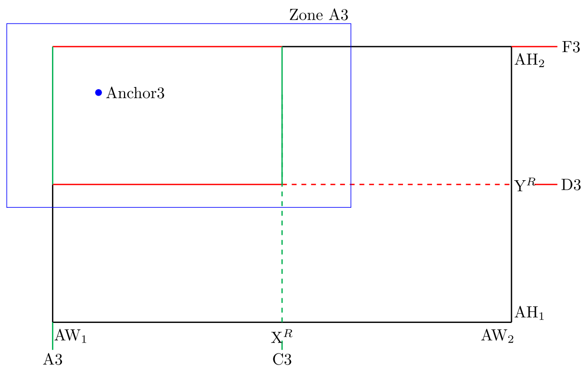


FIGURE 7. Third case inside the anchorZone() method.

3) CASE 3 OF THE anchorZone() METHOD FIRST STAGE

The third case is displayed in Figure 7 where the condition of the anchor coordinate for the x-axis point is less than or equal to the x-axis range calculated earlier using (3) and the y-axis point is more than the y-axis range which calculated earlier using (4). The A3 denotes the start value of the anchor zone range width and the value is given by the start value of the area width. The C3 indicates the end value of the anchor zone range width and the value is given by the start value of the area width plus the x-axis range value. The D3 represents the start value of the anchor zone range height and the value given by the y-axis range value. Lastly, the F3 signifies the end value of the anchor zone range height and the value given by the end value of the area height. After getting all of the zone value, the A3 will be the axisX1 which is the start points and the C3 will be the axisX2 which is the endpoint of the anchor zone at the x-axis. The D3 will be the axisY1 which is the start point and the F3 will be the axisY2 which is the endpoint of the anchor zone at the y-axis.

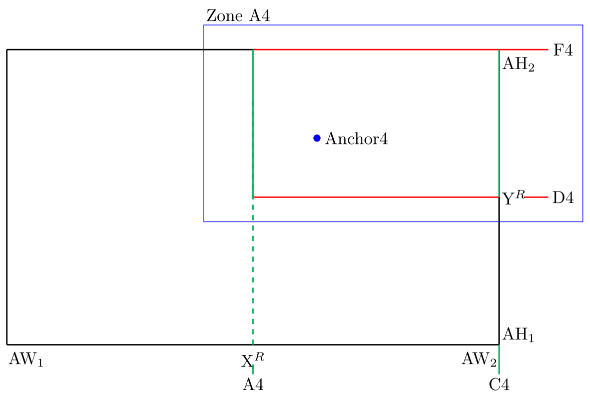


FIGURE 8. Fourth case inside the anchorZone() method.

4) CASE 4 OF THE anchorZone() METHOD FIRST STAGE

The fourth case is displayed in Figure 8 where the condition of the anchor coordinate for the x-axis point is more than the x-axis range calculated earlier using (3) and the y-axis point is also more than the y-axis range which calculated earlier using (4). The A4 denotes the start value of the anchor zone range width and the value is given by the x-axis range value. The C4 indicates the end value of the anchor zone range width and the value is given by the end value of the area width. The D4 represents the start value of the anchor zone range height and the value given by the y-axis range value. Lastly, the F4 signifies the end value of the anchor zone range height and the value given by the end value of the area height. After getting all of the zone value, the A4 will be the axisX1 which is the start point and the C4 will be the axisX2 which is the endpoint of the anchor zone at the x-axis. The D4 will be the axisY1 which is the start point and the F4 will be the axisY2 which is the endpoint of the anchor zone at the y-axis.

B. CREATING THE anchorZone() METHOD SECOND STAGE

Algorithm 4 shows the anchorZone() method for the second stage algorithm. The second stage explained how to locate the position of the anchor zone range if there are 2 anchors inside the first stage zone. The algorithm starts by assigning a parameter S as the status to check whether the anchor is inside the same zone or not. The S parameter is assigned value as 2. The condition is placed by set if the parameter S is 1, the anchors are in a different zone and if the parameter S is 2, the anchors are in the same zone. For each of the parameter j inside the tS minus 1 array, the iL here will be set to 0. Then for each of the parameter k inside the tS where the parameter k is the parameter j plus 1, several conditions will be checked here. If the axisX1 of parameter j is equal to axisX1 of parameter k and axisX2 of parameter j is equal to axisX2 of parameter k and axisY1 of parameter j is equal to axisY1 of parameter k and axisY2 of parameter j is equal to axisY2 of parameter k, the iL will be checked whether parameter j contains inside the iL. If parameter j does not inside the iL, the parameter j will be added inside the iL. If there is already parameter j inside the iL the iL will add parameter k.

1) THE ZONE POSITION CHARACTERIZATION

The zone position characterization takes place inside the area based on the position of the anchor. Based on Figure 5–8, the anchor those are used as the examples is to be placed in the first quarter of the field area. The situation also applied to the other part of the field area based on the anchor coordinates.

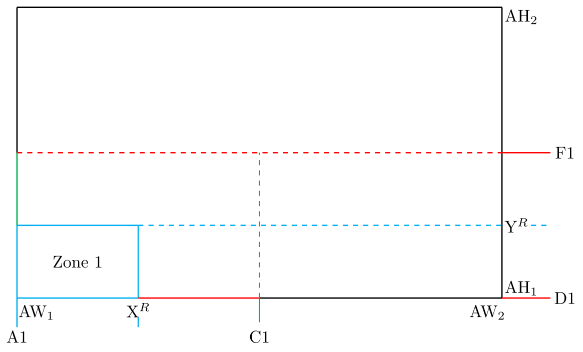


FIGURE 9. Zone 1 division example.

Figure 9 shows the example of the division that is classified as zone 1, where it might be happened to have the anchor inside. For each of the parameter m inside the iL, if the parameter m inside the iL equals 1(which means inside the zone 1), the x-axis range is equal to the axisX2 denote as C1 minus the axisX1 denote as A1 and the result is divide by 2. The y-axis range is equal to the axisY2 denote as F1 minus the axisY1 denote as D1 and the result is divided by 2. The value of the axisX1, axisX2, axisY1, and axisY2 come from the parameter m inside the iL.

Figure 10 shows the example of the division that is classified as zone 2, where it might be happened to have the anchor inside. If the parameter m inside the iL equals 2(which means inside the zone 2), the x-axis range is equal to the axisX2 denote as C1 minus the axisX1 denote as A1 then divide by 2, and the result complete by adding the axisX1 which denote as A1. The y-axis range is equal to the axisY2 denote as F1 minus the axisY1 denote as D1 and the result is divided by 2. The value of the axisX1, axisX2, axisY1, and axisY2 are coming from the parameter m inside the iL.

Figure 11 shows the example of the division that is classified as zone 3, where it might be happened to have the anchor inside. If the parameter m inside the iL equals 3(which means inside the zone 3), the x-axis range is equal to the axisX2 denote as C1 minus the axisX1 denote as A1 then divide by 2. The y-axis range is equal to the axisY2 denote as F1 minus the axisY1 denote as D1 divide by 2 and the result complete by adding the axisY1 which denotes as D1. The value of the axisX1, axisX2, axisY1, and axisY2 are coming from the parameter m inside the iL.

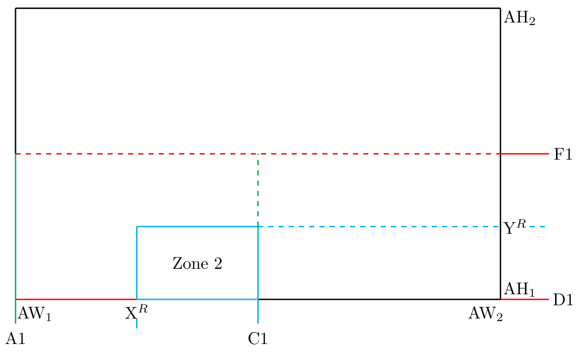


FIGURE 10. Zone 2 division example.

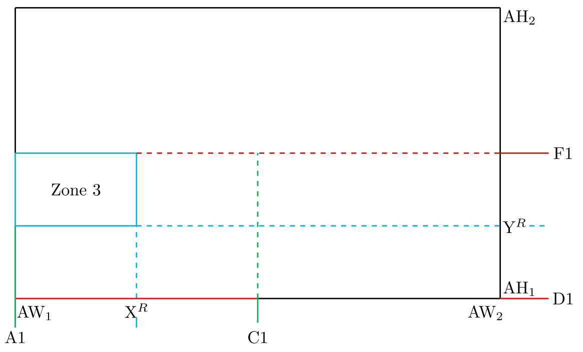


FIGURE 11. Zone 3 division example.

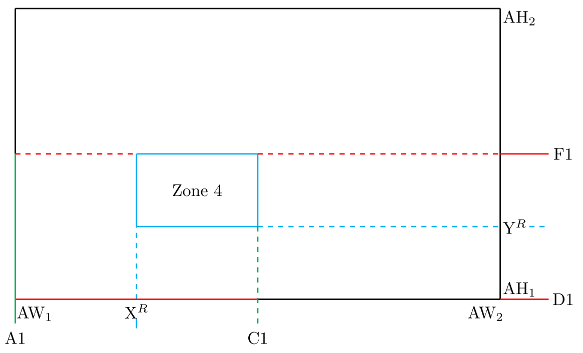


FIGURE 12. Zone 4 division example.

Figure 12 shows the example of the division that is classified as zone 4, where it might be happened to have the anchor inside. If the parameter m inside the iL equals 4(which means inside the zone 4), the x-axis range is equal to the axisX2 denote as C1 minus the axisX1 denote as

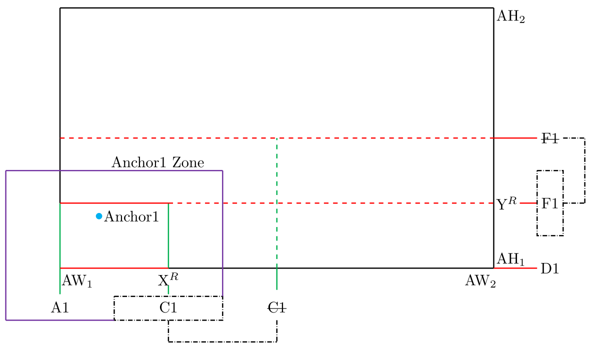


FIGURE 13. **The first case of the zone position characterization situation.**

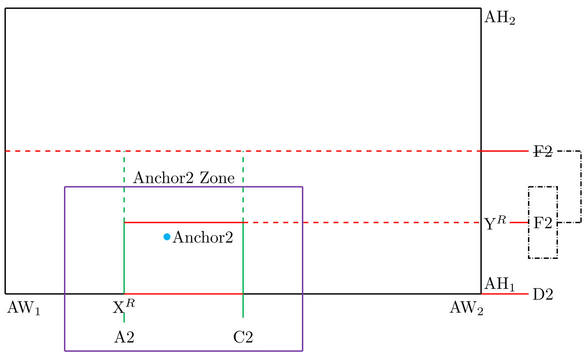


FIGURE 14. **The second case of the zone position characterization situation.**

A1 then divide by 2, and the result complete by adding the axisX1 which denote as A1. The y-axis range is equal to the axisY2 denote as F1 minus the axisY1 denote as D1 divide by 2 and the result complete by adding the axisY1 which denotes as D1. The value of the axisX1, axisX2, axisY1, and axisY2 come from the parameter m inside the iL.

2) THE ZONE POSITION CHARACTERIZATION CASE 1

The first case displayed in Figure 13 happened at line 24-33 inside the anchorZone() Algorithm second stage showed by Algorithm 4. The case starts by checking the Anchor1 coordinate for x-axis point that came from the parameter m from the iL is less than or equal to the x-axis range and the y-axis point that came from the parameter m from the iL is also, lesser than or equal to the y-axis range. If it is true, the A1 is used to represent the start value of the Anchor1 zone range width and the value is given by the axisX1. The C1 indicates the end value of the Anchor1 zone range width and the value is given by the previous axisX2 denote as C1 minus the axisX1 denote as A1 then divide by 2 and the result complete by adding the axisX1 which denote as A1. Now, the previous C1 is no longer the end value and the recent end value denotes as the C1 in the box. The D1 represents the start value of the Anchor1 zone range height and the value is given by the axisY1. Lastly, the F1 signifies the end value of the Anchor1 zone range height and the value is given by the previous axisY2 denote as F1 minus the axisY1 denote as D1 then divide by 2, and the result complete by adding the axisY1 which denote as D1. Now, the previous F1 is no longer the end value and the recent end value denotes as the F1 in the box.

After getting all of the zone value, the A1 will be the axisX1 which is the start point and the C1 will be the axisX2 which is the endpoint of the Anchor1 zone at the x-axis. The D1 will be the axisY1 which is the start point and the F1 will be the axisY2 which is the endpoint of the Anchor1 zone at the y-axis. The value of the axisX1, axisX2, axisY1, and axisY2 come from the parameter m inside the iL. The parameter m from the iL will be used to store in zone array which is declared as the first zone.

3) THE ZONE POSITION CHARACTERIZATION CASE 2

The second case displayed by Figure 14 happened at line 34-43 inside the anchorZone() Algorithm second stage showed by Algorithm 4. The case starts by checking the Anchor2 coordinate for x-axis point that came from the parameter m from the iL is more than the x-axis range and the y-axis point that came from the parameter m from the iL is lesser than or equal to the y-axis range. If it is true, the A2 is used to represent the start value of the Anchor2 zone range width and the value is given by the x-axis range. The C2 indicates the end value of the Anchor1 zone range width and the value is given by the axisX2. The D2 represents the start value of the Anchor2 zone range height and the value is given by the axisY1. Lastly, the F2 signifies the end value of the Anchor2 zone range height and the value is given by the previous axisY2 denote as F2 minus the axisY1 denote as D2 then divide by 2, and the result complete by adding the axisY1 which denote as D2. Now, the previous F2 is no longer the end value and the recent end value denotes the F2 in the box.

After getting all of the zone value, the A2 will be the axisX1 which is the start point and the C2 will be the axisX2 which is the endpoint of the Anchor2 zone at the x-axis. The D2 will be the axisY1 which is the start point and the F2 will be the axisY2 which is the endpoint of the Anchor2 zone at the y-axis. The value of the axisX1, axisX2, axisY1, and axisY2 come from the parameter m inside the iL. The parameter m from the iL will be used to store in zone array which is declared as the second zone.

4) THE ZONE POSITION CHARACTERIZATION CASE 3

The third case displayed in Figure 15 happened at lines 44-53 inside the anchorZone() Algorithm second stage showed by Algorithm 4. The case starts by checking the Anchor3 coordinate for x-axis point that came from the parameter m from the iL is less than or equal to the x-axis range and the y-axis point that came from the parameter m from the iL is more than the y-axis range. If it is true, the A3 is used to represent the start value of the Anchor3 zone range

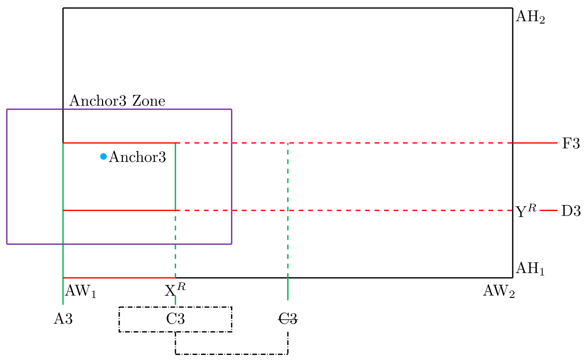


FIGURE 15. The third case of the zone position characterization situation.

width and the value is given by the axisX1. The C3 indicates the end value of the Anchor3 zone range width and the value is given by the previous axisX2 denote as C3 minus the axisX1 denote as A3 then divide by 2 and the result complete by adding the axisX1 which denote as A3. Now, the previous C3 is no longer the end value and the recent end value denotes the C3 in the box. The D3 represents the start value of the Anchor3 zone range height and the value is given by the y-axis range. Lastly, the F3 signifies the end value of the Anchor3 zone range height, and the value is given by the axisY2.

After getting all of the zone value, the A3 will be the axisX1 which is the start point and the C3 will be the axisX2 which is the endpoint of the Anchor3 zone at the x-axis. The D3 will be the axisY1 which is the start point and the F3 will be the axisY2 which is the endpoint of the Anchor3 zone at the y-axis. The value of the axisX1, axisX2, axisY1, and axisY2 come from the parameter m inside the iL. The parameter m from the iL will be used to store in zone array which is declared as the third zone.

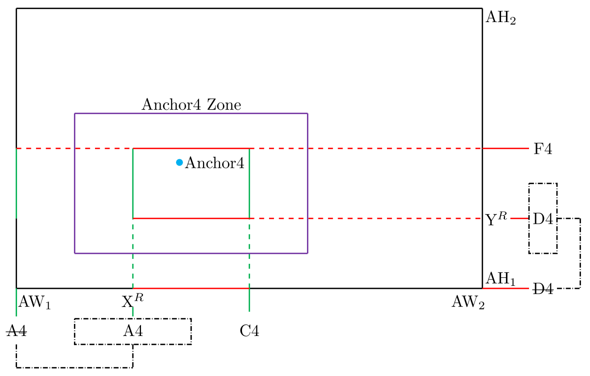


FIGURE 16. The fourth case of the zone position characterization situation.

5) THE ZONE POSITION CHARACTERIZATION CASE 4

The fourth case displayed in Figure 16 happened at line 54-63 inside the anchorZone() Algorithm second stage showed by Algorithm 4. The case starts by checking the Anchor4 coordinate for x-axis point that came from the parameter m from the iL is more than the x-axis range and the y-axis point that came from the parameter m from the iL is also more than the y-axis range. If it is true, the A4 is used to represent the start value of the Anchor4 zone range width and the value is given by the x-axis range. Now, the previous A4 is no longer the start value and the recent start value denotes as the A4 in the box. The C4 indicates the end value of the Anchor4 zone range width and the value is given by the axisX2. The D4 represents the start value of the Anchor4 zone range height and the value is given by the y-axis ranges. Lastly, the F4 signifies the end value of the Anchor4 zone range height, and the value is given by the axisY2. Now, the previous D4 is no longer the end value and the recent end value denotes the D4 in the box.

After getting all of the zone value, the A4 will be the axisX1 which is the start point and the C4 will be the axisX2 which is the endpoint of the Anchor4 zone at the x-axis. The D4 will be the axisY1 which is the start point and the F4 will be the axisY2 which is the endpoint of the Anchor4 zone at the y-axis. The value of the axisX1, axisX2, axisY1, and axisY2 come from the parameter m inside the iL. The parameter m from the iL will be used to store in zone array which is declared as the fourth zone.

C. CREATING THE anchorZone() METHOD THIRD STAGE

Algorithm 5 shows the anchorZone() method for the third stage algorithm. The third stage gives details on how to summarize the position of the anchor whether it is in the same zone or not. For each of the parameter m in tS minus 1, the latest result of the axisX1 from parameter m will be saved as parameter a. Same goes to the axisX2 where it saved as parameter b, axisY1 saved as parameter c, and axisY2 saved as parameter d. After that, the algorithm will check for each parameter n in tS where parameter n is equal to parameter m plus 1. Here the axisX1 for parameter n will be saved as parameter e, axisX2 as parameter f, axisY1 as parameter g, and axisY2 as parameter h.

After that, all of the parameters will be check. If parameter a is not equal to parameter e and parameter b is not equal to parameter f and parameter c is not equal to parameter g and finally, parameter d is not equal to parameter h, the status will be 1 which means they are in the different zone. After that, if parameter a is equal to parameter e and parameter b is equal to parameter f and parameter c is equal to parameter g and finally, parameter d is equal to parameter h, the status will be 2 which means they are in the same zone. The loop at line 1 will break once this has been recognized.

D. THE connectTo() ALGORITHM FOR NODES AND NODES WITH ANGLE

The connectTo() method is applied to find out which anchor the nodes and angle nodes connect to inside the build cloud. Since the character for the nodes is moderately similar to the nodes inside the DDSPAN Algorithm shown in Algorithm 6, the connectTo() method for nodes inside FCDV-Hop applied

Algorithm 6 DDSPAN Algorithm

```
(TN: total number of nodes, a: area)
 1: Initialize parameters
 2: Distribute node using uniform distribution
 3: for each object $i \in TN$ do
 4:     generate coordinate randomly
 5: for each object $i \in TN$ do
 6:     for each object $j \in TN$ do
 7:         if i = j
 8:             continue line 6
 9:         $\delta_{ij} = \sqrt{(X[i] - X[j])^2 + (Y[i] - Y[j])^2}$
10: for each object $i \in \delta$ where $(1 \leq i \leq 20)$ do
11:     for each object $i \in \delta_i$ do
12:         if $\delta_{ij} > 1$
13:             $\epsilon = 2$
14: for each object $i \in TN$ do
15:     $\eta_m$ = Integer.MAX_VALUE where $(1 \leq m \leq 3)$
16:     for each object $i \in TN$ do
17:         apply findMin()
18:     $\beta_{iq} = \eta_m$ where $(0 \leq q \leq 2)$
19: for each object $i \in \beta$ where $(1 \leq i \leq 20)$ do
20:     for each object $j \in \beta_i$ do
21:         $\Omega = \beta_{ij}$
22:         $\xi = \delta_{i\Omega}$
23:         count no of $j$ connect to each $i$
24:         if count $\geq 1$ and $\leq 4$
25:             $i$ = anchor
26: for each object $k \in TN$ do
27:     apply matrixMultiply()
28: for each object $i \in TN$ do
29:     $\lambda = \frac{\sqrt{(X[i]-X[j])^2+(Y[i]-Y[j])^2}}{\alpha \times TN}$
```

based on the DDSPAN algorithm. The connectTo() method for nodes with angle showed by Algorithm 7. First, the angle node is connecting to each of the anchors. For each parameter i, the Ʊ for each of the parameter k will be equal to *\$* of parameter k. *\$* represents the minimum value of the angle nodes. The Ʊ will hold the value inside the minimum array for the angle nodes. After that, the fl for each parameter k, where fl signifies as the minimum distance for each angle nodes are equal to δ for each angle nodes of parameter k that hold the previous minimum array value.

Then, the number of angle nodes that connect to one anchor is calculated. For each of the parameter i, if c of parameter k is equal to 4, the condition for *\$* will be check. If array *\$* of parameter k is not equal to i, then the array will be changed to be equal to i. If c of parameter k is more than 4 or less than 4, the condition for *\$* will be check. If array *\$* of parameter k is not equal to i, then the position of angle nodes will be check. If the condition of the X point for each parameter k of the angle nodes is equal to x point of the anchor or the Y point for each parameter k of the angle nodes are equal to y point of the anchor, the *\$* of the parameter k is equal to i.

Algorithm 7 connectTo() Algorithm for Node With Angle

```
 1: Initialize Ʊ: Minimum between Angle Node; £: Minimum Value; c: counter
 2: for each object $i$ where $(1 \leq i \leq 4)$, $(1 \leq k \leq 4)$ and (j = 0) do
 3:     $\text{Ʊ}_{k_i} = \text{£}_{k_{ij}}$
 4:     $\text{Æ}_{k_i} = \delta_{\text{ɣ}N_{k_{i\text{Ʊ}}}}$
 5: for each object $i$ where $(1 \leq i \leq 4)$, $(1 \leq k \leq 4)$ and (j = 0) do
 6:     if $c_k = 4$
 7:         if $\text{£}_{k_{ij}} \neq i$
 8:             $\text{£}_{k_{ij}} = i$
 9:     if $(c_k > 4)$ @ $(c_k < 4)$
10:         if $\text{£}_{k_{ij}} \neq i$
11:             if $(X_{k_{\text{ɣ}N_i}} = X_{A_i})$ @ $(Y_{k_{\text{ɣ}N_i}} = Y_{A_i})$
12:                 $\text{£}_{k_{ij}} = i$
```

E. THE matrixMultiply() ALGORITHM FOR NODES AND NODES WITH ANGLE

Algorithm 8 matrixMultiply() Algorithm

```
 1: Initialize Λ: Matrix
 2: for each object $i \in TN$ do
 3:     $A = \beta_{i0}$
 4:     $B = \beta_{i1}$
 5:     $C = \beta_{i2}$
 6:     Initialize parameters
 7:     $\Lambda 1_{ef} = 2 \times (X_A - X_C)$ where $(0 \leq e \leq 1)$ and $(f = 0)$ then Remove A and continue to B
 8:     $\Lambda 1_{ef} = 2 \times (Y_A - Y_C)$ where $(0 \leq e \leq 1)$ and $(f = 1)$ then Remove A and continue to B
 9:     left = $\Lambda 1_{00} \times \Lambda 1_{11}$
10:     right = $\Lambda 1_{01} \times \Lambda 1_{10}$
11:     determinant = left - right
12:     temp = $\Lambda 1_{00}$
13:     $\Lambda 1_{00} = \Lambda 1_{11}$
14:     $\Lambda 1_{11}$ = temp
15:     $\Lambda 1_{01} = -\Lambda 1_{01}$
16:     $\Lambda 1_{10} = -\Lambda 1_{10}$
17:     $\Lambda 1_{ef} = \Lambda 1_{ef}/\text{determinant}$ where $(0 \leq e \leq 1)$ and $(f = 0)$
18:     $\Lambda 1_{ef} = \Lambda 1_{ef}/\text{determinant}$ where $(0 \leq e \leq 1)$ and $(f = 1)$
19:     $\Lambda 2_e = X_A^2 - X_C^2 + Y_A^2 - Y_C^2 + \delta_{iC}^2 - \delta_{iA}^2$ where $(0 \leq e \leq 1)$ then Remove A and continue to B
20:     $\Lambda 3_{ie} = (\Lambda 1_{e0} \times \Lambda 2_0) + (\Lambda 1_{e1} \times \Lambda 2_1)$ where $(0 \leq e \leq 1)$
```

The matrixMultiply() method is used to find the estimated coordinate of the angle nodes. Since the character for the nodes are moderately similar to the nodes inside the DDSPAN Algorithm shown in Algorithm 6, the matrixMultiply() method for nodes inside FCDV-Hop applied based on the matrixMultiply() shown by Algorithm 8.

Algorithm 9 matrixMultiply() Algorithm for Node With Angle

```
1:  for each object $i \in ɣN_k$, where $(1 \le k \le 4)$ do
2:      $A_k = £_{ki0}$
3:      $B_k = £_{ki1}$
4:      $C_k = £_{ki2}$
5:      $\Lambda 1_{k_{ef}} = 2 * \left(X_{ɣN_{k_{A_k}}} - X_{ɣN_{k_{C_k}}}\right)$ where $(0 \le e \le 1)$ and $(f = 0)$ then Remove A and continue to B
6:      $\Lambda 1_{k_{ef}} = 2 * \left(Y_{ɣN_{k_{A_k}}} - Y_{ɣN_{k_{C_k}}}\right)$ where $(0 \le e \le 1)$ and $(f = 0)$ then Remove A and continue to B
7:      $\text{left}_k = \Lambda 1_{k_{00}} * \Lambda 1_{k_{11}}$
8:      $\text{right}_k = \Lambda 1_{k_{01}} * \Lambda 1_{k_{10}}$
9:      $\text{determinant}_k = \text{left}_k - \text{right}_k$
10:     $\text{temp}_k = \Lambda 1_{k_{00}}$
11:     $\Lambda 1_{k_{00}} = \Lambda 1_{k_{11}}$
12:     $\Lambda 1_{k_{11}} = \text{temp}_k$
13:     $\Lambda 1_{k_{01}} = \Lambda 1_{k_{01}}$
14:     $\Lambda 1_{k_{10}} = \Lambda 1_{k_{10}}$
15:     $\Lambda 1_{k_{ef}} = \Lambda 1_{k_{ef}}$ /$\text{determinant}_k$ where $(0 \le e \le 1)$ and $(f = 0)$
16:     $\Lambda 1_{k_{ef}} = \Lambda 1_{k_{ef}}$ /$\text{determinant}_k$ where $(0 \le e \le 1)$ and $(f = 1)$
17:     $\Lambda 1_{k_{ef}} = X^2_{ɣN_{k_{A_k}}} - X^2_{ɣN_{k_{C_k}}} + Y^2_{ɣN_{k_{A_k}}} - Y^2_{ɣN_{k_{C_k}}} + \delta_{ɣN^2_{iC_k}} - \delta_{ɣN^2_{iA_k}}$ where $(0 \le e \le 1)$ then Remove A and continue to B
18:     $\Lambda 3_{k_{ie}} = (\Lambda 1_{ke0} * \Lambda 2_{k_0}) + (\Lambda 1_{ke1} * \Lambda 2_{k_1})$ where $(0 \le e \le 1)$
```

Algorithm 9 shows the matrixMultiply() method algorithm for the node with angle. For each of the parameter i of the angle nodes, 3 parameters are assigned. The parameter A, parameter B, and parameter C are assigned to hold the three shortest minimum distance found by the findMin() algorithm. The parameter A will hold the $, followed by the parameter B and the parameter C. A matrix array is used to calculate the 2 × 2 matrix to get the determinant value. The 2 × 2 matrix is inversed and the matrix array now will be dividing by the determinant that already calculated before. After that, the second matrix array will be used to do the calculation. Finally, the third matrix array is used to calculate the first and second matrix array.

V. PERFORMANCE ANALYSIS

The performance analysis technique employed for this research is the discrete-event simulation. The main component of the simulation algorithm was the anchors, the nodes, and the angle nodes. The anchors work as the station where will be receiving both nodes information while both of the nodes will give all of their information to the nearest anchors. The total number of experiments conducted was ten times for anchor and both nodes. The simulation parameters used are shown in Table 2. FCDV-Hop control parameters for the number of nodes generated that are used in the simulation is fixed because the nodes used are for 4 number of anchors.

TABLE 2. Main parameters of the simulation model.

Input Simulation Parameter	Values
Total number of nodes	36
Number of Nodes with Angles	16
Anchor Proportion	0.1
AH_1, AW_1	0
AH_2, AW_2	10
Dimensions	100unit^2
Radius	<1

The simulation developed using Java programming language, which used the Eclipse IDE for Java Developers version Kepler Release software. The experiments deployed on the Alienware M15X laptop with Intel(R) Core (TM) i7 processor and Random-Access Memory (RAM) of 4.00 GB RAM. The operating system was the Windows 7 Home Premium.

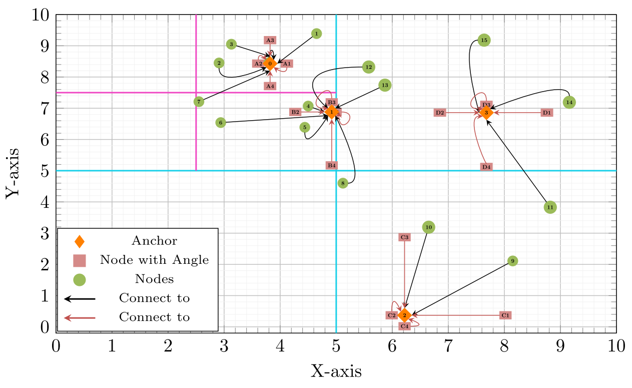


FIGURE 17. Distribution of 36 nodes scatter inside 10 × 10 units square shape area.

A. SIMULATION ENVIRONMENT

The topology for the second proposed idea consists of 10 units X 10 units in a square-shaped area. Figure 17 illustrates the topology for FCDV-Hop where the numbers of nodes are 36 nodes that were randomly distributed inside the area. The number of nodes used in this simulation is fixed. The 4 closest nodes between the nodes and the nodes with an angle will be selected to connect to the anchor. The maximum number of nodes that connect to the anchor is 4 to save energy consumption during the flood.

The transmission power used in FCDV-Hop is 2dBm for 10 units which mean as per unit, 0.2dBm will be used. From the distance of the nodes and the nodes with angles calculated, the transmission power calculated using (5).

$$P_r = c\frac{P_t}{d^a} \Leftrightarrow d = \sqrt[a]{\frac{cP_t}{P_r}} \tag{5}$$

B. SIMULATION SCENARIOS

FCDV-Hop nodes distribution will be depending on the anchor nodes. As summarized in Table 2, the simulation scenarios are pictured in Figure 17 where the numbers of nodes are 36 nodes which consist of 16 of the nodes, 16 of the nodes with angles, and 4 of the anchors. The number of nodes used in this simulation is fixed. All of the nodes distributed inside the area of 10 units times 10 units. The nodes distributed inside the anchor zone randomly but at the same time depend on the size of the anchor zone. The angle nodes are generated randomly but based on the location of the anchor and also the size of the anchor zone. Angles nodes are restricted to 16 based on the anchor proportion which is 0.1. That means every anchor node may generate 4 number of angles nodes inside the anchor zone. FCDV-HOP managed to place the anchor inside a region and used the position of the anchor and also its region to generate the nodes and the nodes with angles inside the area. This proposed idea controls the flood of the node base on its anchor.

Each of the anchors will own a specific region to ease the anchor chooses which node that will be connected to be the neighbor inside their zone. After all of the nodes are distributed, the nodes and the nodes with an angle will begin to calculate the distance to the anchor and the results are stored for the shortest distance nodes.

TABLE 3. Result of the nodes and the nodes with angle connection to the anchor.

Nodes	Connect to Anchor	Angle Nodes	Connect to Anchor
0	0	A1	0
1	0	A2	0
2	0	A3	0
3	0	A4	0
4	1	B1	1
5	1	B2	1
6	1	B3	1
7	0	B4	1
8	1	C1	2
9	2	C2	2
10	2	C3	2
11	3	C4	2
12	1	D1	3
13	1	D2	3
14	3	D3	3
15	3	D4	3

Table 3 shows the results of the connection between the nodes and the angle nodes to the anchor. For the nodes, the connection mapped using the blue arrow whereas the nodes with angles mapped using the orange arrow inside Figure 17.

Nodes 0, 7, and 12 denoted by the orange square connected to the same node which is node 15 and this will point to node 15 as the anchor. This case is similar for nodes 16, 1, and node 13. For node 2, the only node that is connecting to it is node 3, which automatically will be a pointer as the anchor.

FCDV-Hop will choose the nodes inside the anchor zone and within the specified zone. For each anchor, 4 selected nodes will be chosen by the angle of 90°, 180°, 270°, and 360°. The angle is used to control the position of the nodes with angle and to make sure nodes with angle placed in the right zone. In this event, for each time the simulation runs, it is starting to calculate the nodes with angle distance from the anchor node coordinate. The distance between all of the nodes with angle is also calculated.

The anchors' positions are represented by the area of x1 and x2 for the x-axis, and y1 and y2 for the y-axis after generated randomly. As 4 anchors were generated in the area of 10 units times 10 units, all of the anchors will form their zone. For the first round of developing the zone, all of the anchors must form a different area scale. If the results showed that the anchor produced the same section after comparing each of the anchor's sections, for example on the x-axis that is shown by anchor A and B, the section will be divided for both the x-axis and y-axis. The division of both the x-axis and y-axis must be made at the same time to avoid the affected area having a redundant location. The process for locating each anchor zone will be completed once the results showed the x-axis and y-axis position for each anchor are different.

This simulation scenario is presented by using 16 nodes and 16 nodes with an angle to state how the simulation is done. This is because if a bigger number of both nodes are used, it will crowd space, and the explanations may be confusing. The nodes' placement and labeling also will be confusing. Figure 18 shows the distribution of nodes for anchor C and anchor D. As both of the anchors are in a different zone, the nodes and the angle nodes are generated directly inside the area. For anchor C, the nodes connected are nodes 9 and 10 and the angle nodes are C1, C2, C3, and C4 which sum as 6 nodes connect to anchor C. Anchor C choose the angle nodes C1, C2, C3, and C4 to be connected to as the angle nodes are much closer to the anchor compared to the nodes. Anchor D having 7 nodes connected to it which are nodes 11, 14, and 15, and the angle nodes of D1, D2, D3, and D4. Anchor C chooses the angle nodes to be connected to as the angle nodes are much closer to the anchor compared to the nodes the same as the situation of anchor C. Figure 19 shows the distribution of nodes for anchor A and anchor B. As both of the anchors are in the same zone, the nodes and the angle nodes zone are dividing again inside the earlier zone. In this case, just for one divide, anchor A and B are already in a different zone. If they are still inside the same zone, it will divide until they are not in the same zone. For anchor A,

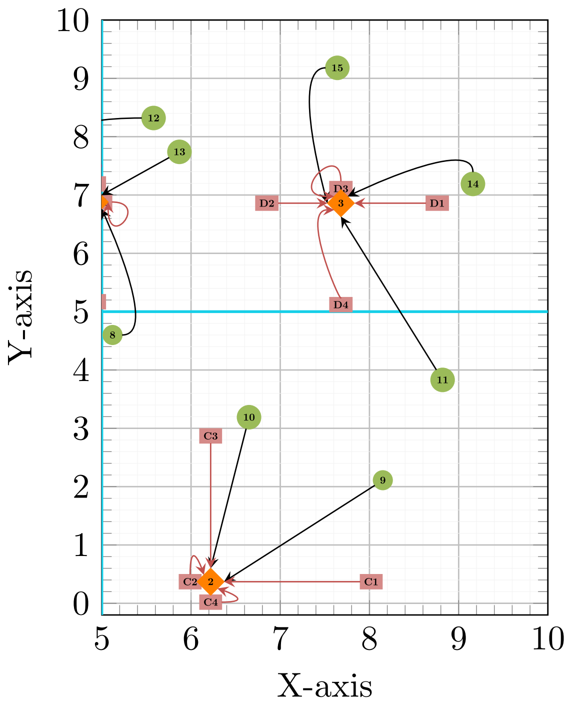


FIGURE 18. The close up for anchor C and D distribution.

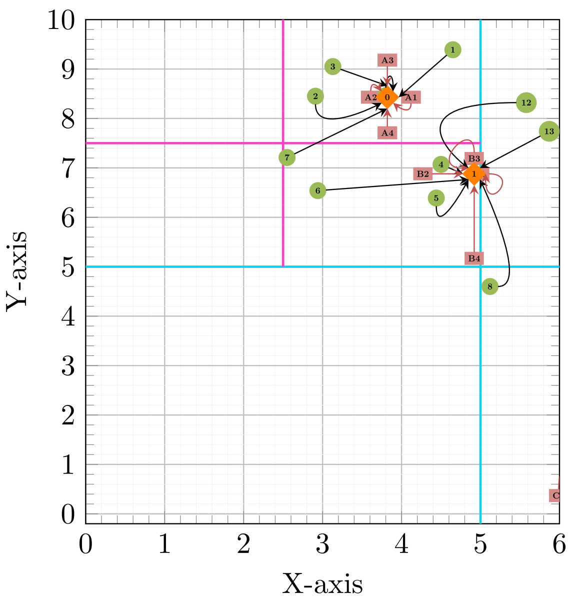


FIGURE 19. The close up for anchor A and B distribution.

the nodes connected are the nodes 0, 1, 2, 3, and 7 and the angle nodes are A1, A2, A3, and A4 which sum as 9 nodes connect to anchor A. Anchor A choose the angle nodes A1, A2, A3, and A4 to be connected to as the angle nodes are much closer to the anchor compared to the nodes. Anchor B having 10 nodes connect to it which are nodes 4, 5, 6, 8, 12, and 13 and the angle nodes of B1, B2, B3, and B4. Anchor B chooses the angle nodes to be connected to as the angle nodes are much closer to the anchor compared to the nodes the same as the situation of anchor A.

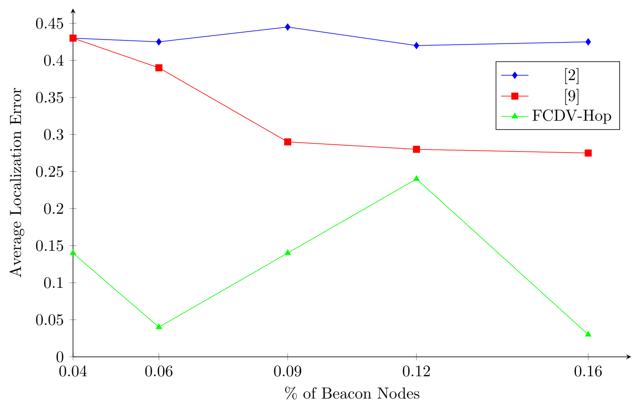


FIGURE 20. The average localization error against the percentage of beacon nodes comparison between DV-Hop, the benchmark [9] and FCDV-Hop.

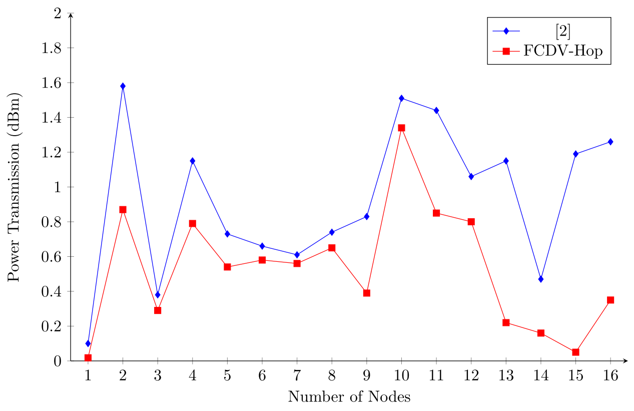


FIGURE 21. The power transmission against the node number.

VI. RESULTS AND DISCUSSION

Figure 20 shows the result of the average location error of nodes varies from the percentage of the beacon nodes between the [2] algorithm and FCDV-Hop. The nodes are used here to make the comparison where 0.04% of beacon nodes mean 4 beacon nodes are used over 100 total numbers of nodes. FCDV-Hop clearly shows the lowest localization error percentage compares to [2]. However, at the percentage 0.06 until 0.12 of the beacon nodes, the result is increasing than the other percentage of beacon nodes. This is due to the high density of the nodes inside the area where the number of the beacons is getting higher making the nodes localization becoming more complex thus affect the calculation of the localization errors. Figure 21 shows the result of the power transmission varies from the nodes number between [2] and FCDV-Hop. The nodes used in this result are the nodes with an angle. FCDV-Hop clearly shows that the power transmissions used are lower than the DV-Hop for each of the nodes. Several of the nodes for FCDV-Hop power transmission are higher than the other nodes because the nodes with angle placement are far from the anchor nodes compare to the DV-Hop nodes. Figure 22 shows the localization error varies from the node number between [2] and FCDV-Hop.

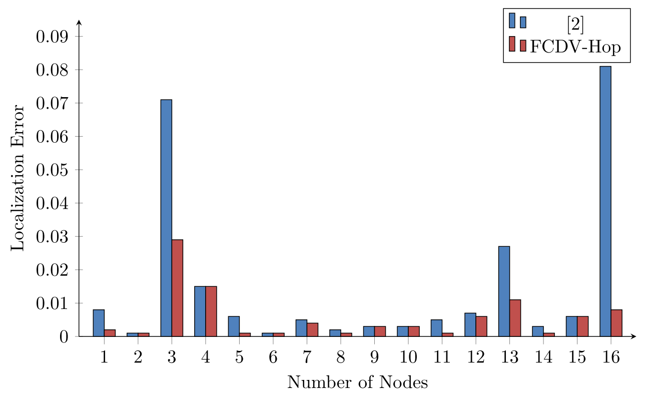


FIGURE 22. The localization error against the node number.

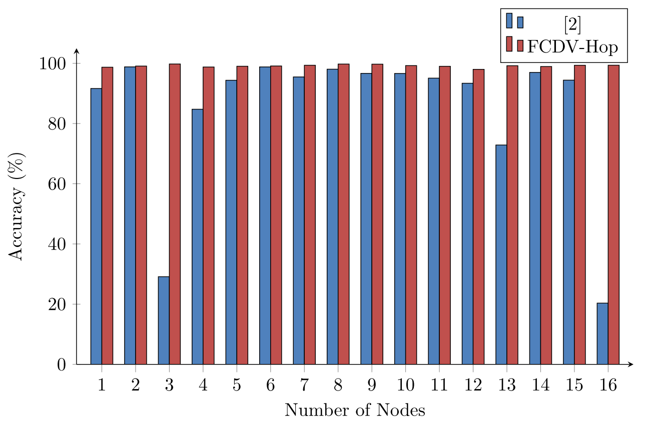


FIGURE 23. The percentage of accuracy against the node number.

The nodes used in this result are the nodes with an angle. FCDV-Hop clearly shows that the localization errors are mostly higher than the DV-Hop for most of the nodes because the nodes with angle placement are generated inside the zone of the anchor compared to the DV-Hop nodes where it is generated randomly inside the area without considering the position of the anchor. As the nodes with angles are generated inside the anchor zone; it is possible that the placement of the nodes with angles must come to a condition that the area to be distributed is limited. The localization error for FCDV-Hop also shown much more stable results compared to the DV-Hop. Figure 23 shows the result of the percentage of accuracy varies from the nodes number between [2] and FCDV-Hop. The nodes used in this result are the nodes with an angle. FCDV-Hop clearly shows a higher accuracy compares to [2]. This is due to the placement of the nodes with an angle that is much well-located compared to the DV-Hop nodes considered that the nodes with angle have their area to be located compared to the larger area of the DV-Hop nodes have to be distributed.

VII. SUMMARY

This paper describes the flooding control of the node's placement for the localization of the DV-Hop algorithm. There are several requirements for a positioning algorithm. First, it must be able to distribute in a large network with low memory and low bandwidth nodes which move inside a topology to a base station in the hop by hop movement manners. Second, the amount of node to node communication and computation power used must be minimized as for the low signaling and computation complexity during the nodes flooding them information. Third, the positioning system would be able to work if the network was already disconnected. Finally, the algorithm must provide an absolute positioning that enables a unique name-space. The placement of nodes is limited inside the region of the anchor nodes. The simulation environment setup comes with parameters such as the radius, area, number of nodes, types of nodes, the anchor proportion, and the value of the regions. The results are discussed and show a series of experiments to examine the positioning algorithm proposed and comparing the result with the existing works. The enhancement that has been made in this research is the reengineering of the region area for the sensor nodes placement inside the localization area. The equal division of nodes area idea solved the problem for controlling the flood movement problems in WSN. This approach not only produced a better result in the flood movement, but it also increases the network lifetime and reduces the power consumption while minimizing the localization error.

One of the future works that can be done is by considering the hop as an inaccurate interference. This approach can amend the determination of the hop itself for the hop lengths measurements. The other future work to be considered is towards the sensor nodes' flood movements. The flood of the sensor nodes' movements can be controlled by considering how much energy is used or energy left for each sensor node. This approach can prevent the sensor nodes from moving towards the low energy sensor nodes. The proposed ideas also can be extended to real WSN applications instead of only using the simulation for future work.

REFERENCES

[1] M. Potnuru and P. Ganti, "Wireless sensor network: Issues, challenges and survey of solutions," in *Proc. IEEE*. Champaign, IL, USA: Univ. of Illinois Urbana, 2003, pp. 2–18.

[2] D. S. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proc. IEEE Global Telecommun. Conf.*, San Antonio, TX, USA, Nov. 2001, pp. 2926–2931.

[3] A. Gupta, M. Gupta, and A. Nayyar, "Approaches for combating delay and achieving optimal path efficiency in wireless sensor networks," *J. Comput. Sci. Inf. Technol.*, vol. 3, no. 5, pp. 105–111, May 2014.

[4] I. Amundson and X. D. Koutsoukos, "A survey on localization for mobile wireless sensor networks," in *Mobile Entity Localization Tracking GPS-less Environnments*, vol. 5801. Berlin, Germany: Springer, 2009, pp. 235–254, doi: 10.1007/978-3-642-04385-7_16.

[5] C. Bolin and Z. Zengwei, "LLSiWSN: A new range-free localization algorithm for large scale wireless sensor netwoks," in *Proc. Int. Conf. Bus. Comput. Global Informatization*, Shanghai, China, Jul. 2011, pp. 408–411.

[6] Y. Chen, L. Shu, M. Li, Z. Fan, L. Wang, and T. Hara, "The insights of DV-based localization algorithms in the wireless sensor networks with duty-cycled and radio irregular sensors," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kyoto, Japan, Jun. 2011, pp. 1–6.

[7] T. Hu, "A range-free wireless sensor networks localization algorithm," in *Proc. IEEE Int. Conf. Eng. Ind.*, Jeju, South Korea, Nov./Dec. 2011, pp. 1–5.

[8] S. Kumar, T. Singhal, and T. Kapil, "Enhanced composite approach with mobile Beacon shortest path to solve localization problem in wireless sensor network," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 12, pp. 7579–7585, Dec. 2010.

[9] Y. Meng, J. Chen, Y. Wen, and H. Zhao, "The four corners DV-hop localization algorithm for wireless sensor network," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Changsha, China, Nov. 2011, pp. 1733–1738.

[10] J. Du and X. Yan, "An improved DV_HOP algorithm used to failure localization on power grid cables," in *Proc. 9th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci.*, Hong Kong, Aug. 2010, pp. 337–341.

[11] S. Li, X. Kong, D. Lowe, and H.-G. Ryu, "Wireless sensor network localization with autonomous mobile beacon by path finding," in *Proc. Int. Conf. Inf. Sci. Appl.*, Suwon, South Korea, May 2012, pp. 1–6.

[12] P. Kumar, A. Chaturvedi, and M. Kulkarni, "Geographical location based hierarchical routing strategy for wireless sensor networks," in *Proc. Int. Conf. Devices, Circuits Syst. (ICDCS)*, Coimbatore, India, Mar. 2012, pp. 9–14.

[13] S. Yang, J. Yi, and H. Cha, "HCRL: A Hop-Count-Ratio based localization in wireless sensor networks," in *Proc. 4th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, San Diego, CA, USA, Jun. 2007, pp. 31–40.

[14] N. Sharma and A. Nayyar, "A comprehensive review of cluster based energy efficient routing protocols for wireless sensor networks," *Int. J. Appl. Innov. Eng. Manage.*, vol. 3, no. 1, pp. 441–453, Jan. 2014.

[15] F. Benbadis, T. Friedman, M. Dias de Amorim, and S. Fdida, "GPS-free-free positioning system for wireless sensor networks," in *Proc. 2nd IFIP Int. Conf. Wireless Opt. Commun. Netw. (WOCN)*, Dubai, UAE, 2005, pp. 541–545.

[16] Y. Dai, J. Wang, and C. Zhang, "Improvement of DV-hop localization algorithms for wireless sensor networks," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, Chengdu, China, Sep. 2010, pp. 1–4.

[17] H. Ahn and J. Hong, "DV-hop localization algorithm with multi-power beacons under noisy environment," in *Proc. 3rd Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Dalian, China, Jun. 2011, pp. 7–12.

[18] H.-Q. Cheng, H.-K. Wang, and H. Wang, "Research on centroid localization algorithm that uses modified weight in WSN," in *Proc. Int. Conf. Netw. Comput. Inf. Secur.*, Guilin, China, May 2011, pp. 287–291.

[19] A. Jahangiry, R. Ahmadi, and M. Mirnia, "Wireless Sensor Networks Localization with Improvement in Energy Consumption," in *Proc. IEEE Int. Conf. Comput. Sci. Netw. Technol.*, Harbin, China, Dec. 2011, pp. 2174–2177.

[20] P. Kristalina and G. Hendrantoro, "Improved range-free localization methods for wireless sensor networks," in *Proc. Int. Conf. Electr. Eng. Informat.*, Bandung, Indonesia, Jul. 2011, pp. 1–6.

[21] S. Li, D. Lowe, X. Kong, and R. Braun, "Wireless sensor network localization algorithm using dynamic path of mobile beacon," in *Proc. 17th Asia Pacific Conf. Commun.*, Sabah, Malaysia, Oct. 2011, pp. 344–349.

[22] S. Li, X. Kong, and D. Lowe, "Dynamic path determination of mobile beacons employing reinforcement learning for wireless sensor localization," in *Proc. 26th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Fukuoka, Japan, Mar. 2012, pp. 760–765.

[23] X. Yingxi, G. Xiang, S. Zeyu, and L. Chuanfeng, "WSN node localization algorithm design based on RSSI technology," in *Proc. 5th Int. Conf. Intell. Comput. Technol. Autom.*, Zhangjiajie, China, Jan. 2012, pp. 556–559.

[24] Q. Zhang, J. Wang, and C. Jin, "A distributed node localization algorithm for WSNs based on the Newton method," in *Proc. 7th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Wuhan, China, Sep. 2011, pp. 1–5.

[25] V. Vivekanandan and V. W. S. Wong, "Concentric anchor beacon localization algorithm for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 2733–2744, Sep. 2007.

[26] K. Chen, Z.-H. Wang, M. Lin, and M. Yu, "An improved DV-hop localization algorithm for wireless sensor networks," in *Proc. IET Int. Conf. Wireless Sensor Netw. (IET-WSN)*, Singapore, 2010, pp. 1557–1561.

[27] K. Liu, X. Yan, and F. Hu, "A modified DV-hop localization algorithm for wireless sensor networks," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, Shanghai, China, Nov. 2009, pp. 511–514.

[28] F. Yassine and H. Safa, "A hybrid DV-hop for localization in large scale wireless sensor networks," in *Proc. 6th Int. Conf. Mobile Technol., Appl. Syst. Mobility*, New York, NY, USA, 2009, pp. 48–53.

[29] G. Teng, K. Zheng, and W. Dong, "Adapting mobile beacon-assisted localization in wireless sensor networks," *Sensors*, vol. 9, no. 4, pp. 2760–2779, Apr. 2009.

[30] W. Liu, E. Wang, Z. Chen, and L. Wang, "An improved DV-hop localization algorithm based on the selection of beacon nodes," *J. Converg. Inf. Technol.*, vol. 5, no. 9, pp. 157–164, Nov. 2010.

[31] N. Yu, J. Wan, Q. Song, and Y. Wu, "An improved DV-Hop localization algorithm for wireless sensor networks," in *Proc. IEEE Int. Conf. Inf. Acquisition*, Chengdu, China, Aug. 2006, pp. 266–269.

[32] G. Wei, X. He, B. Zhou, and W. Wei, "A meshless method for DV-HOP localization," in *Proc. 2nd Int. Conf. Signal Process. Syst.*, Dalian, China, Jul. 2010, pp. 629–632.

[33] A. Nayyar and R. Singh, "A comprehensive review of simulation tools for wireless sensor networks (WSNs)," *J. Wireless Netw. Commun.*, vol. 5, no. 1, pp. 19–47, Jan. 2015.

[34] L. Tang, W. Chai, X. Chen, and J. Tang, "Research of WSN localization algorithm based on moving beacon node," in *Proc. 3rd Pacific–Asia Conf. Circuits, Commun. Syst. (PACCS)*, Wuhan, China, Jul. 2011, pp. 1–5.

[35] L. Shu, Y. Zhang, L. T. Yang, Y. Wang, M. Hauswirth, and N. Xiong, "TPGF: Geographic routing in wireless multimedia sensor networks," *Telecommun. Syst.*, vol. 44, nos. 1–2, pp. 79–95, Jun. 2010.

[36] Y. Hu, Z. Shan, and H. Yu, "Research on improved DV-HOP localization algorithm based on the ratio of distances," in *Internet of Things* (Communications in Computer and Information Science), vol. 312. Berlin, Germany: Springer, 2012, pp. 118–125, doi: 10.1007/978-3-642-32427-7_17.

[37] Y. Liu, H. Luo, C. Long, and N. Zhou, "Improved DV-hop localization algorithm based on the ratio of distance and path length," *J. Inf. Comput. Sci.*, vol. 9, no. 7, pp. 1875–1882, Jul. 2012.

[38] M. Ramazany and Z. Moussavi, "Localization of nodes in wireless sensor networks by MDV-Hop algorithm," *ARPN J. Syst. Softw.*, vol. 2, no. 5, pp. 166–171, May 2012.

[39] X. Chen and B. Zhang, "Improved DV-Hop node localization algorithm in wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 8, no. 8, pp. 1–7, Aug. 2012.

[40] A. A. Agashe and R. S. Patil, "An optimum DV Hop localization algorithm for variety of topologies in wireless sensor networks," *Int. J. Comput. Sci. Eng.*, vol. 4, no. 6, pp. 957–961, Jun. 2012.

[41] P. Tinh and M. Kawai, "Distributed range-free localization algorithm based on self-organizing maps," *EURASIP J. Wireless Commun. Netw.*, vol. 2010, no. 1, pp. 1–9, Nov. 2009, doi: 10.1155/2010/692513.

[42] D. Zhang, F. Liu, L. Wang, and Y. Xing, "DV-hop localization algorithms based on centroid in wireless sensor networks," in *Proc. 2nd Int. Conf. Consum. Electron., Commun. Netw. (CECNet)*, Yichang, China, Apr. 2012, pp. 3216–3219.

[43] Y. Zhang, K. Wang, S. Yuan, H. Yang, Z. Chen, and L. Ge, "Research of WSN node localization algorithm based on weighted DV-HOP," in *Proc. 24th Chin. Control Decis. Conf. (CCDC)*, Taiyuan, China, May 2012, pp. 3826–3829.

[44] B. Wang, G. Wu, S. Wang, and L. T. Yang, "Localization based on adaptive regulated neighborhood distance for wireless sensor networks with a general radio propagation model," *IEEE Sensors J.*, vol. 14, no. 11, pp. 3754–3762, Nov. 2014.

[45] S. Lee, B. Koo, and S. Kim, "RAPS: Reliable anchor pair selection for range-free localization in anisotropic networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1403–1406, Aug. 2014.

[46] S. Savazzi, M. Nicoli, F. Carminati, and M. Riva, "A Bayesian approach to device-free localization: Modeling and experimental assessment," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 1, pp. 16–29, Feb. 2014.

[47] C. Wu, Z. Yang, and Y. Liu, "Smartphones based crowdsourcing for indoor localization," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 444–457, Feb. 2015.

[48] S. Kianoush, S. Savazzi, F. Vicentini, V. Rampa, and M. Giussani, "Device-free RF human body fall detection and localization in industrial workplaces," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 351–362, Apr. 2017.

[49] J.-F. Huang, G.-Y. Chang, and G.-H. Chen, "A historical-beacon-aided localization algorithm for mobile sensor networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 6, pp. 1109–1122, Jun. 2015.

[50] A. El Assaf, S. Zaidi, S. Affes, and N. Kandil, "Robust ANNs-based WSN localization in the presence of anisotropic signal attenuation," *IEEE Wireless Commun. Lett.*, vol. 5, no. 5, pp. 504–507, Oct. 2016.

[51] F. Shahzad, T. R. Sheltami, and E. M. Shakshuki, "DV-maxHop: A fast and accurate range-free localization algorithm for anisotropic wireless networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 9, pp. 2494–2505, Sep. 2017.

[52] S. Zaidi, A. El Assaf, S. Affes, and N. Kandil, "Accurate range-free localization in multi-hop wireless sensor networks," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3886–3900, Sep. 2016.

[53] S. Phoemphon, C. So-In, and N. Leelathakul, "Fuzzy weighted centroid localization with virtual node approximation in wireless sensor networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4728–4752, Dec. 2018.

[54] Z. Wang, H. Zhang, T. Lu, and T. A. Gulliver, "A grid-based localization algorithm for wireless sensor networks using connectivity and RSS rank," *IEEE Access*, vol. 6, pp. 8426–8439, 2018, doi: 10.1109/ACCESS.2018.2804381.

[55] I. Ullah, J. Chen, X. Su, C. Esposito, and C. Choi, "Localization and detection of targets in underwater wireless sensor using distance and angle based algorithms," *IEEE Access*, vol. 7, pp. 45693–45704, 2019, doi: 10.1109/ACCESS.2019.2909133.

[56] N. Saeed, A. Celik, T. Y. Al-Naffouri, and M.-S. Alouini, "Localization of energy harvesting empowered underwater optical wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 5, pp. 2652–2663, May 2019.

[57] N. Saeed, A. Celik, M.-S. Alouini, and T. Y. Al-Naffouri, "Performance analysis of connectivity and localization in multi-hop underwater optical wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2604–2615, Nov. 2019.

[58] G. Kumar, R. Saha, M. K. Rai, R. Thomas, and T.-H. Kim, "A lattice signcrypted secured localization in wireless sensor networks," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3949–3956, Sep. 2020.

[59] Y. Zhao, J. Xu, J. Wu, J. Hao, and H. Qian, "Enhancing camera-based multimodal indoor localization with device-free movement measurement using WiFi," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1024–1038, Feb. 2020.

[60] W. Fang and G. Yang, "Improvement based on DV-hop localization algorithm of wireless sensor network," in *Proc. Int. Conf. Mech. Sci., Electric Eng. Comput. (MEC)*, Jilin, China, Aug. 2011, pp. 2421–2424.

[61] J. Jiang, G. Han, H. Xu, L. Shu, and M. Guizani, "LMAT: Localization with a mobile anchor node based on trilateration in wireless sensor networks," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Houston, TX, USA, Dec. 2011, pp. 1–6.

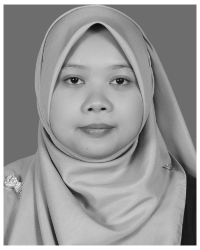
[62] V. K. Chaurasiya, R. L. Lavavanshi, S. Verma, G. C. Nandi, and A. K. Srivastava, "Localization in wireless sensor networks using directional antenna," in *Proc. IEEE Int. Advance Comput. Conf.*, Patiala, India, Mar. 2009, pp. 131–134.

[63] G. Yu, F. Yu, and L. Feng, "A three dimensional localization algorithm using a mobile anchor node under wireless channel," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Hong Kong, Jun. 2008, pp. 477–483.

[64] Y. Zhu, B. Zhang, F. Yu, and S. Ning, "A RSSI based localization algorithm using a mobile anchor node for wireless sensor networks," in *Proc. Int. Joint Conf. Comput. Sci. Optim.*, Sanya, China, Apr. 2009, pp. 123–126.

[65] P. D. Patil and R. S. Patil, "Performance analysis of DV-hop localization using Voronoi approach," *Int. J. Modern Eng. Res.*, vol. 3, no. 4, pp. 1958–1964, Apr. 2013, doi: 10.6084/M9.FIGSHARE.1065532.

[66] H. Wu and R. Gao, "An improved method of DV-Hop localization algorithm," *J. Comp. Inf. Syst.*, vol. 7, no. 7, pp. 2293–2298, Jul. 2011.



AZYYATI ADIAH ZAZALI (Fellow, IEEE) received the bachelor's and master's degrees in computer science from Universiti Putra Malaysia (UPM), in 2010 and 2015, respectively, where she is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology. Her research interests include resource management, wireless sensor networks, simulation, and modeling.

SHAMALA K. SUBRAMANIAM (Member, IEEE) received the bachelor's, master's, and Ph.D. degrees in computer science from Universiti Putra Malaysia (UPM), in 1996, 1999, and 2002, respectively. She is currently a Professor with the Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, UPM. Her research interests include computer networks, simulation and modeling, and scheduling and real-time systems.

ZURIATI AHMAD ZUKARNAIN (Member, IEEE) received the bachelor's and master's degrees in physics and education from Universiti Putra Malaysia (UPM), in 1997 and 2000, respectively, and the Ph.D. degree in quantum computing and communication from the University of Bradford, U.K., in 2005. At the faculty, she taught several courses for undergraduate students, such as data communication and networks, distributed systems, mobile and wireless, network security, computer architecture, and assembly language. For postgraduate students, she taught a few courses, such as the advanced distributed and research method. She has been an Academic Staff with the Faculty of Computer Science and Information Technology, UPM, since 2001. She was the Head of the Department of Communication Technology and Networks, from 2006 to 2011. She was also the Head of the Section of High-Performance Computing, Institute of Mathematical Research, UPM, from 2012 to 2015. Her areas of specialization include computer networks, distributed systems, mobile and wireless, network security, quantum computing, and quantum cryptography. She is a member of the IEEE Computer Society.