

# A Localized Adaptive Architecture For Dynamic Wireless Sensor Networks

by

Sajjad Hussain

16201075

Muntasir Ahmed

17101190

Kishanu Roy Joy

18101705

Md. Mehedi Hasan

17201149

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science

Department of Computer Science and Engineering  
Brac University  
October 2020

© 2020. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

## Student's Full Name & Signature:



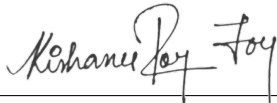
---

Sajjad Hussain  
16201075



---

Muntasir Ahmed  
17101190



---

Kishanu Roy Joy  
18101705



---

Md. Mehedi Hasan  
17201149

# Approval

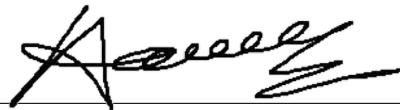
The thesis titled “A Localized Adaptive Architecture For Dynamic Wireless Sensor Networks” submitted by

1. Sajjad Hussain (16201075)
2. Muntasir Ahmed (17101190)
3. Kishanu Roy Joy (18101705)
4. Md. Mehedi Hasan (17201149)

of Summer, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 09, 2020.

## Examining Committee:

Supervisor:  
(Member)



---

Amitabha Chakrabarty, PhD  
Associate Professor  
Department of Computer Science and Engineering  
Brac University

Program Coordinator:  
(Member)



---

Md. Golam Rabiul Alam, PhD  
Associate Professor  
Department of Computer Science and Engineering  
Brac University

Head of Department:  
(Chair)

**Prof. Mahbub Majumdar**  
**Chairperson**  
**Dept. of Computer Science & Engineering**  
**Brac University**

---

Mahbubul Alam Majumdar, PhD  
Professor and Chairperson  
Department of Computer Science and Engineering  
Brac University

# Abstract

IoT is seen as the next big thing in the world to come. Previously people were less connected with the internet. But in recent years, IoT has seen substantial research and innovation in many technology areas, and uses of the internet are increasing day by day. In the near foreseeable future, the vast number of devices connected together will result in highly complex networks that would cause an exponential increase in computation times, latency and power consumption. In this paper, we take a modular approach to dealing with the issue. We propose a Weight-based Adaptive Neighbor Localization (WANLoc) Algorithm that creates a highly optimized network based on localized network architecture that is network independent. This means that the sensor nodes in this network do not need to communicate with any cluster head or sink in order to function. In addition, WANLoc is intended to function in a dynamic wireless sensor network where nodes enter and exit the network regularly. This also means that it is exceptionally good at handling node failures. Nodes in this network only communicate with their immediate neighbors, maintained in a set of dynamic tables designed to reduce lookup times when deciding whom to forward data to. This also means that nodes can now be configured with minimum transmission capabilities, saving a lot of precious energy in the process.

**Keywords:** WANLoc, Wireless Sensor Networks; Self-Adaptive; Internet of Things; Network Architecture; Dynamic; Routing Algorithm; Network Topology.

## Acknowledgement

Firstly, we would like to praise our Lord for whom we have managed to finish our thesis even in the current global pandemic. We were able to do our best and successfully complete it within time.

Secondly, we would like to convey our gratitude to our supervisor Amitabha Chakrabarty, PhD for his time, effort and contribution throughout the whole phase of our thesis work as well as to write this document. From the very beginning to the end, he has guided and inspired us to move forward to reach our goal. He listened to every problem we faced and gave us valuable insights, pulling us out of deadlocks with his visionary ideas which worked brilliantly.

Thirdly, we are also very much thankful to our parents, friends, and well-wishers who have helped us directly or indirectly while conducting our research and continuing our work.

Furthermore, we would also like to acknowledge the assistance we received from a number of resources over the internet, journals and papers, specially from the works of our fellow researchers.

Finally, we would like to thank BRAC University for giving us the opportunity to conclude the thesis and for giving us the chance to complete our Bachelor degree.

# Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Nomenclature	ix
<b>1 Introduction</b>	<b>1</b>
1.1 Influence of IoT . . . . .	2
1.2 Future of IoT . . . . .	2
1.3 Aims and Objective . . . . .	4
1.4 Thesis Overview . . . . .	5
<b>2 Literature Review</b>	<b>6</b>
2.1 Mesh Networks . . . . .	6
2.2 Hierarchical Networks . . . . .	8
<b>3 WANLoc: Weight based Adaptive Neighbor Localization Algorithm</b>	<b>14</b>
3.1 Localization . . . . .	16
3.1.1 Geometric proof . . . . .	16
3.2 Weighted Nodes . . . . .	17
3.2.1 Function Comparison . . . . .	18
<b>4 Implementation and Result</b>	<b>22</b>
4.1 WANLoc Parameters . . . . .	22
4.2 Dormant State . . . . .	24
4.3 Mature State . . . . .	26
4.4 Result Analysis . . . . .	29
4.5 Advantages . . . . .	31
4.5.1 Complexity . . . . .	31

4.5.2 Scalability . . . . .	32
<b>5 Conclusion</b>	<b>34</b>
5.1 <i>Application</i> . . . . .	34
<b>Bibliography</b>	<b>35</b>

# List of Figures

1.1	IoT Device Usage Over the Year . . . . .	2
1.2	Network Architecture Complexity. . . . .	3
1.3	Complexity. . . . .	3
2.1	Broadcast Route Request from source node 1 to destination node 9. . . . .	7
2.2	Route Reply from destination node 9 to source node 1. . . . .	8
2.3	DSDV Routing Protocol in Network. . . . .	9
2.4	Shortest Path Finding Algorithm CGSR. . . . .	10
2.5	Broadcast Route Request from source node 1 to destination node 9. . . . .	11
2.6	Route Reply from destination node 9 to source node 1. . . . .	11
2.7	Propagation of Query message. . . . .	12
2.8	Node's height updated as a result of update message. . . . .	12
3.1	Flow Chart for WANLoc Algorithm. . . . .	15
3.2	Equidistant points in 1D. . . . .	16
3.3	Equidistant points in 2D. . . . .	17
3.4	Equidistant points in 3D. . . . .	17
3.5	Weight vs. Global Radius. . . . .	18
3.6	Weight assignment in WANLoc. . . . .	19
3.7	Weighted SNs . . . . .	20
3.8	Heatmap based on Weight . . . . .	20
3.9	Weight assignment with multiple SINKs . . . . .	21
3.10	Heatmap of a Multi-SINK network . . . . .	21
4.1	WANLoc simulation result on 400 SN network. . . . .	29
4.2	WANLoc simulation result on 200 SN network. . . . .	30



# List of Tables

4.1	SN parameters in WANLoc . . . . .	23
4.2	Logical Comparison of Algorithms . . . . .	33

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$d$  Working dimension of the WSN

$IoT$  Internet of Things

$LEACH$  Low-Energy Adaptive Clustering Hierarchy

$Loc$  Location array

$N_{max}$  Maximum connectable neighbors

$R$  Global Radius of SN from SINK

$r$  Local radius between two SNs

$SINK$  Sink Node

$SN$  Sensor Node

$W$  Weight of Sensor Node

$WANLoc$  Weight based Adaptive Neighbor Localization Algorithm

$WSN$  Wireless Sensor Network

# Chapter 1

## Introduction

Internet of things (IoT) establishes a network that communicates and exchanges data between themselves and between people and things by the IoT devices for smarter services to the user [1-2]. IoT network methodologies include smart home facilities and infrastructure, smart protection and monitoring, and sophisticated road traffic control and medical emergency response systems [3]. IoT networks are immense in scale and complexity and comprise objects like Radio Frequency Identification (RFID) tags, mobile phones, and sensing devices to obtain data from the environment. These kinds of devices, also entitled sensor nodes, have low computed capability and limited battery life. If a node fails, the nodes around them become 'sensitive-nodes'. What it means is that the amount of load the failed node was supposed to take is going to pass that load to the surrounding nodes. The energy of wireless sensor nodes cannot be replenished, one of the key factors which shall be considered in the design is how to effectively use energy supply and extend the working life of the network [4]. There are diverse types of communication that may subsist among the IoT network and that embrace device to device, human to the device, and contrariwise and device to distributed shortage system. Communication can be inside the same network or among the heterogeneous networks [11]. Additionally, a device to device communication can happen either with human involvement or not. Communication may also be single-hop or multi-hops. Within the single-hop communication, devices connect with one another through a network interface that may be an entry purpose or a base station. Since communication through multi-hops, devices transfer data for each other to attain end-to-end communication with either source or destination device [11]. It is also notable that, on a large scale, the IoT framework contains numerous networks [12]. However, most IoT routing strategies rely on WSN, as it is considered the heart of IoT networks.

Wireless Sensor Networks (WSN) is a key factor in today's world. Apart from the traditional wired networks, they have the obvious advantages of being small, low-cost, low power, and multi-functional sensing devices. These small sensing devices have the capabilities of sensing, computation, self-organizing, and communication known as sensors. Sensor is a tiny device used to sense the condition of its surroundings, gather data, and process it to draw some meaningful information that can be used to recognize the phenomena around its environment. The current routing protocols for WSNs are complicated and demand a considerable use of processing power and memory that is scarce resources within the devices comprising an IoT network [3].

## 1.1 Influence of IoT

The area of IoT has seen a lot of research and advancement in several application regions recently. IoT is considered as the next big thing in the future world. For example, in 2003, 500 million devices were connected with the internet, where the total amount of people is almost 6.3 billion in the world. In percentage only .08% of people connected with the internet in 2003 [5]. So, in 2003 IoT was not a necessary thing as less than 1 device used by 1 person. After 2003, Smart-phones and tablet computers came into the market and the use of internet devices increased in a large number of scales. The use of internet devices increased gradually after 2003. In 2010, the device connected with the internet increased significantly which is almost 12.5 billion and a total population of 6.8 billion. In percentage, almost 183% of devices connected with the internet, and the first time in history per person device is almost 2 [5]. In 2015 almost 25 billion devices connected to the internet [6]. Day by day the use of the device increases significantly which makes IoT very important for the secure use of internet devices.

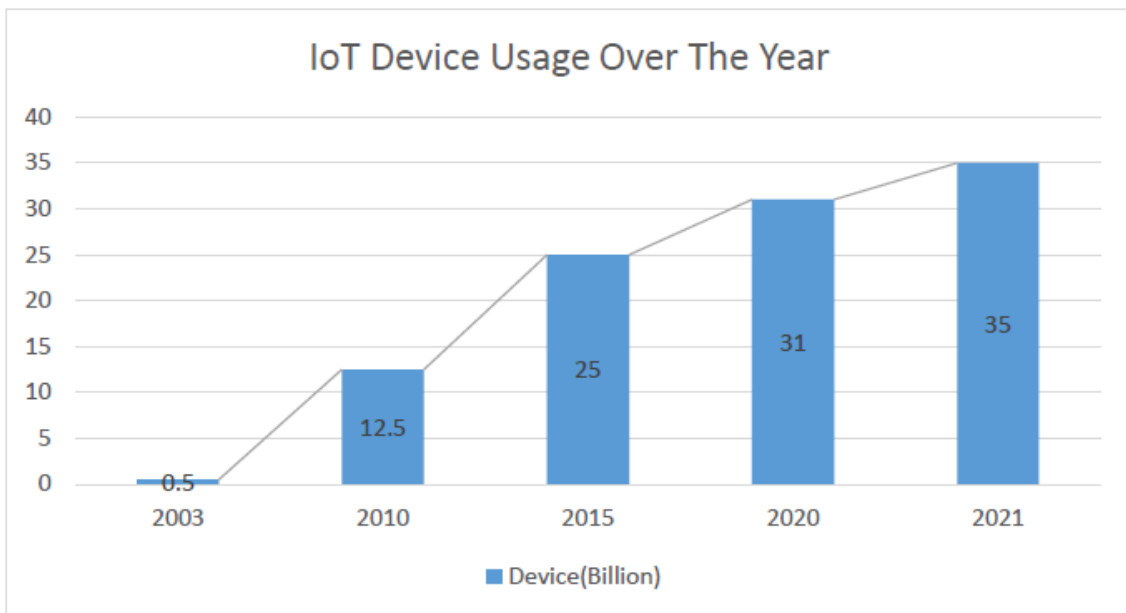


Figure 1.1: IoT Device Usage Over the Year

## 1.2 Future of IoT

According to the article [7], the number of active devices connected with the internet reached 26.66 billion, and every second 127 new devices were added to the internet. The present world (2020) has almost 31 billion connected devices and expects an estimated that it will be more than 35 billion in 2021. On the other hand, the current world population is 7.8 billion. The ratio of the device used is 4:1. The number of devices is increasing almost 4 times higher compared to the global population and this will be continued in the near future. As a result, more cities will be smart cities in the near future. Not only consumers but also cities and companies will try to adopt new technology to save their time and money. As a result, complexity

will increase. Lot of device will connect with each other in a network and if we consider as tree it is getting complicated as it goes on next level. After a certain point complexity will follow a certain line in the ( figure 1.2 , 1.3 ) we can see that what happen if the device increases.

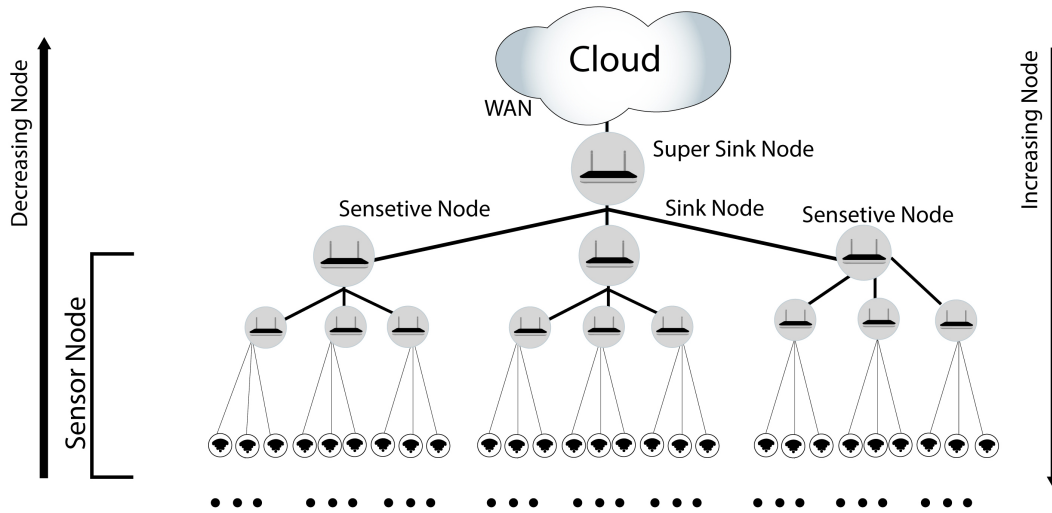


Figure 1.2: Network Architecture Complexity.

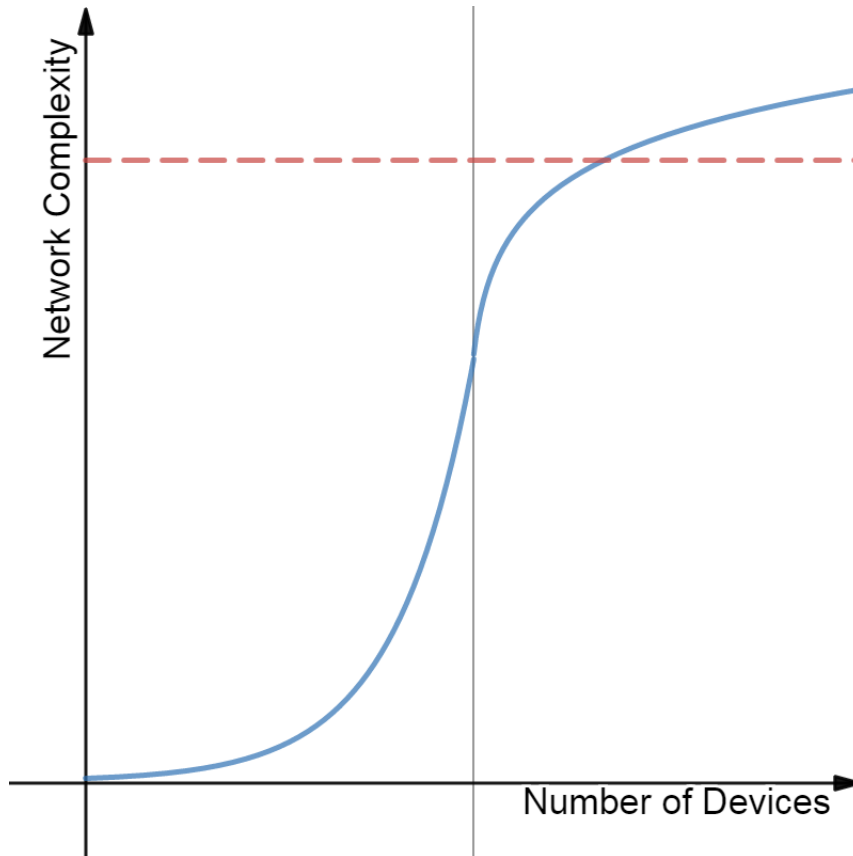


Figure 1.3: Complexity.

These things will add a new dimension to Artificial Intelligence which continues in its path to become a bigger thing. Smart devices collect information data about our habits and needs and using machine learning it gives the preferences of what individual people need. Because of increasing data and big analyzing network speed is very important and the 5G network is the perfect solution.

5G is the fastest network in the present and a faster network helps to gather, analyze, and manage data to a higher degree which makes 5G one of the most demanding things in the future.

In the near foreseeable future, we predict that SNs will also be integrated with drone technology, making future WSNs dynamic which can pose a great challenge for all currently existing network architectures and routing algorithms.

### 1.3 Aims and Objective

In this paper we propose an algorithm that is based on inspirations we got from existing mesh and hierarchical algorithms (notably LEACH), which generate routing paths based on the mesh topology while remaining highly dynamic. The aim is to create paths from a SN to the SINK without the dependence of any cluster heads or be constraint by the distance between the SN and the SINK. In order to do this, the SNs residing in a WSN will only communicate with the SNs in its immediate proximity, known as the neighbor nodes. By carrying out these minimum distanced communications with the neighboring SNs, data will eventually be transferred to the SINK. SNs forward data to other SNs based on the weight of the SNs, a heuristic value that gives the SNs a sense of where to send the data to i.e., where the SINKs are.

WANLoc is also intended to function in a dynamic WSN where SNs enter and exit the network regularly, making it exceptionally good at handling node failures. SNs in this network only communicate with their immediate neighbors, reducing the need to transmit data over long distances. This results in far reduced probability of packet drops and conservation of precious energy. Please note that our aim in this paper is to neither improve any existing algorithm or provide any alternative to them but to design an algorithm can be a blueprint in addressing concerns that the existing algorithms would fail at and in the process demonstrate that our algorithm can out-perform them too by breaking through various constraints.

## 1.4 Thesis Overview

- **Chapter 1** is the introduction of our thesis work. A short summary of our heavy dependency on IoT is mentioned here as well as a short overview of what we are trying to accomplish.
- **Chapter 2** consists of the literature review where we have demonstrated the background study that we have done for this thesis.
- **Chapter 3** is where we take an in depth look at the requirements and the working principle of our proposed algorithm and their proof.
- **Chapter 4** is where we have explained how our algorithm is implemented in detail and analyze its simulation result.
- **Chapter 5** is the conclusion to our paper where discuss possible improvements to our algorithm that can be worked on as well discuss some use cases for it.

# Chapter 2

## Literature Review

In the field of electronic devices which are made up using silicon mostly a transistor is a device that has brought a forcible change in the field of communication. Wireless communication is deemed to be more powerful than before where IoT became a burning topic. However, the existing wireless arrangement procedure among two or more than two gadgets with the help of wave frequency procedure (Radio Frequency) and small distance wireless connection known as Bluetooth seems to be the only solution for these miniature sensor networks where these are simply computer-based network solutions. Besides these solutions, requires more power where the sensor nodes are economical, ground-level devices, mostly operated by external power sources such as lead-acid battery, lithium ion, carbon battery, dry cell battery etc. Since these sensor devices, so-called mini-computers are operated in remote areas; subordinate communication is inappropriate [8].

### 2.1 Mesh Networks

Considering the above mentioned a wireless mesh network (WMN) is a network topology where radio nodes are connected to each other in mesh formation. This concept was introduced in 1970 called Packet Radio Network (PRNET) from the US defense department [8].

There are several types of mesh topology where each performance and operation are different as they are driven by different methods or instructions (algorithms). How they are formed is discussed below:

#### *Infrastructure / Backbone WMNs:*

Realizing from the name we can easily understand that it forms some sort of infrastructure for communication to clients. Here the mesh router plays an important role as it covers the entire area and it is at the same time configured itself. If a link is broken with the sensor node to the router then it provides an alternative route mentioned as a gateway. The important point is that it can be connected to the internet previously which was restricted only by WLAN and Bluetooth based connections [8].

#### *Client WMN:*

It forms a connection similar to a computer to computer connection built upon with help of the internet where each computer can share information or files to another



computer within that network. In the case of sensor nodes, the node that wants to send information generally forms the route in which it is going to send data and make the formation by giving the application to the end-users. The main disadvantage is it cannot access the internet [8].

*Hybrid WMN:*

Combining several infrastructures makes it hybrid. The sensor node can gain access to the internet through the router that resides in its network and also can gain access to the other router and send information to that particular node. Here connectivity is improved though the nodes are dependent on the router [8].

*Optimized Link State Routing (OLSR):*

Link state data is sent over by broadcasting to all the other nodes. It performs information considering three things such as hop count; hop neighbor table and routing table. Using hello packets for the state information. In this protocol Multipoint Relays (MPR) plays an important role. During the data flooding process, node N a subset of neighboring nodes floods the network where other nodes remain idle. When this node information is passed then neighbor nodes receive that message. The main goal in this algorithm is the flooding of information being controlled [9].

*Scalable Routing using Heat Protocol:*

The heat algorithm is propelled by the parameter of hotness. Considering the available path strength and distance the number of nodes, gateways it communicates with the other sensor nodes [9].

*Dynamic Source Routing (DSR):*

It is not a proactive routing protocol rather than a reactive one. Depending on the source of the path it performs its operation in two steps.

They are route discovery and route maintenance. In route discovery, source nodes opt-out for route reserve (cache memory). If the path found sends the data, the PREQ packet is sent. PREQ having an origin address, a target address, route unique number, and record [9]. Figure 2.1, 2.2 shows Broadcast route request and reply destination path

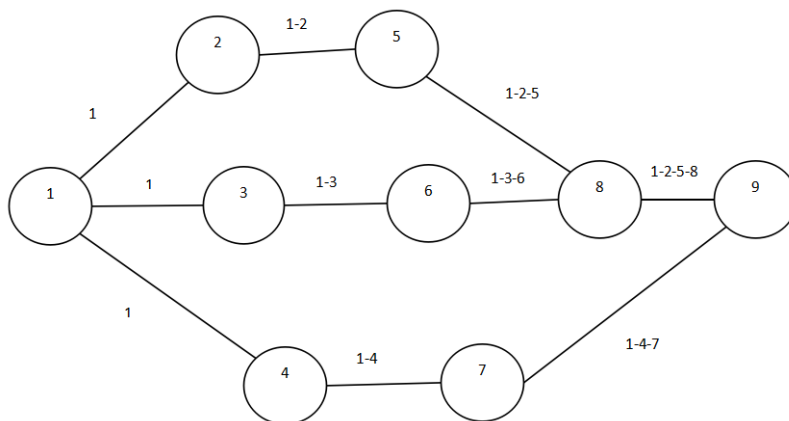


Figure 2.1: Broadcast Route Request from source node 1 to destination node 9.

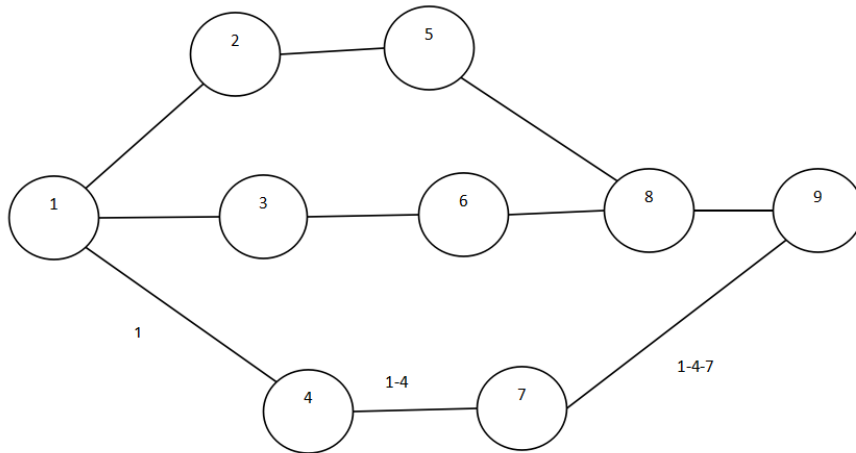


Figure 2.2: Route Reply from destination node 9 to source node 1.

*Link Quality Source Routing (LQSR):*

Another responsive path finding algorithm was developed by Microsoft Research Group. Using parameters such as bounce count, circular travel suspension, packet couple suspension, and Expected Transmission Count (ETX) it develops overexposed link cache instead of route cache. The sink node receiving the Route Request (RREQ) adds the quality metric information which is the crucial part. Since it looks like a “carrying bag” in the back of a person it is a bit energy-consuming compared to other algorithms [9].

*Zone-Based Routing Protocol (ZBRP):*

An area-based hybrid routing protocol is a mixture of both responsiveness and perception. Sensor nodes creating locality is regarded as an area. Minimal distances measured by hop counts are not bigger than the global radius measured from the sink node if the sink node is situated at the center of the topology. Sensor nodes find other sensor nodes from the Neighbor Discovery Protocol (NDP). There are two main features in this ZBRP. One is the Intra Zone Routing Protocol (IARP) where the sensor node performs the path from the source node to the destination by perceiving. The other is Inter-Zone Routing Protocol (IERP) where it only performs when the data is to be passed outside the area the sensor node is residing in [9].

Till now we can see that only infrastructure WMN and hybrid WMN are better for different scenarios [8].

## 2.2 Hierarchical Networks

In hierarchical networks, the sensor nodes by localization form a set of areas where each area is led by cluster heads. These cluster heads perform some dedicated tasks that make them unique from the sensor nodes residing in the network. The fundamental task of a cluster head is performing calculations so that the sensor node power consumption and local calculation overall are brief resulting in less time complexity.

### *Destination Sequenced Distance Vector (DSDV):*

It is based on the Bellman-Ford path finding algorithm where sensor nodes follow a trace board where the minimal path to the destination information is saved along with a number of bounce to reach the destination. In order to get rid of path loops, it follows a sequel number similar to the Bellman-Ford algorithm where it easily differentiates the nodes they have iterated. Whenever the broadcast message is sent from the sink node it has the following information-

1. Sink nodes epicenter (x, y)
2. Hop counts to the sink node
3. Unique sequel number from a particular node to the sink node.

The routing table information, however, performs two things. One is dumping the route information and adding information to the previous board in a fixed time manner [9]. Figure 2.3 shows DSDV routing protocol in network.

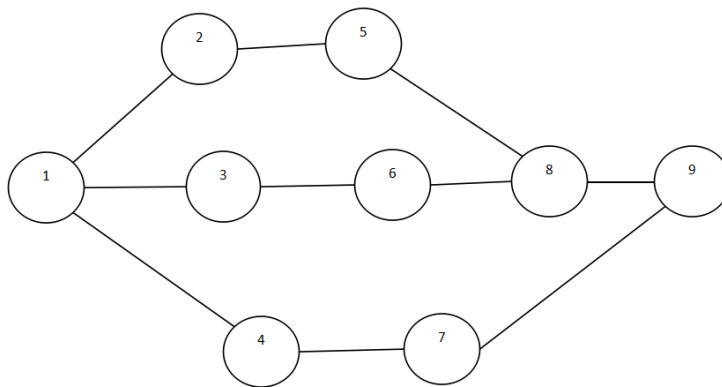


Figure 2.3: DSDV Routing Protocol in Network.

### *Cluster Head Gateway Switch Routing Protocol (CGSR):*

Using destination sequenced distance vector routing (DSDV) protocol in a rank order based. In this type, protocol localization is done by accumulating a number of nodes mentioned as clusters of nodes where each cluster node is led by a leader node known as cluster head thus; rank order (hierarchy) is achieved. Once the topology is formed it imitates the DSDV algorithm where the leader of typical sensor node: cluster head is selected. However, the cluster head is not fixed as it can change based on the scenario it faces such as distance, residual energy, data congestion. As it changes frequently the overall performance becomes bad. Overcoming this problem, Least Cluster Change (LCC) is applied. The job of LCC is to only change the cluster head when a minimal power cluster head comes to interact with a particular sensor node or the sensor node that is about to send the data goes beyond the range of the cluster head that is surrounded by it. The important thing to notice is that along with the routing table it also keeps a separate Cluster Member Table (CMT) for detecting the closest cluster head as well as the path to the destination node (sink node) and the next-hop reaching the cluster head. In the end, the WMN is

separated by more than several cluster heads for load balancing where routing metrics come handy. These metrics help to choose alternative routes and the outcome is global load balancing [9]. Figure 2.4 Shortest path in CGSR shown.

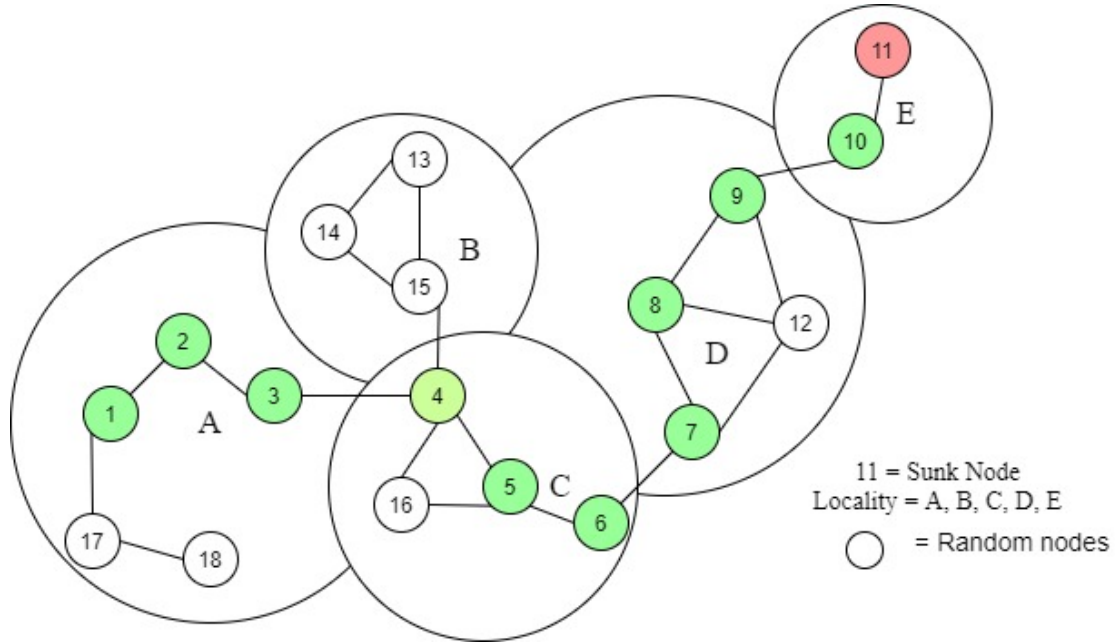


Figure 2.4: Shortest Path Finding Algorithm CGSR.

#### *Ad-Hoc on Demand Distance Vector Routing(AODV):*

Responsive based routing protocol made upon the DSDV routing algorithm. If data is to be sent then and only then the sensor node looks at the routing cache where the destination nodes information is kept. If path information is not available then it sends a Route Request Packet (RRP) to the neighbors. The RRP replied by the neighbors has a origin address, a target address, source sequel number, broadcast spotting number plus the real-time sequel number of origins, and sink node.

Giving a reply from the sink node as the RRP is passed from sensor node to node the path is formed and the sink node knows the path at first that the data is going to come from that sensor node thus AODV gets the path in the shortest and fastest. However, the criteria are by limiting the overwhelming flow of data it achieves efficiency along with the minimal ratio of RRP [9]. Figure 2.5, 2.6 shows Broadcast request and reply path.

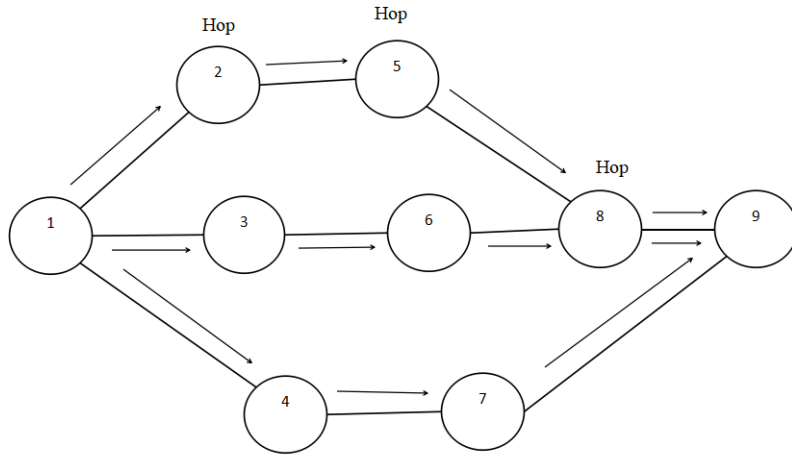


Figure 2.5: Broadcast Route Request from source node 1 to destination node 9.

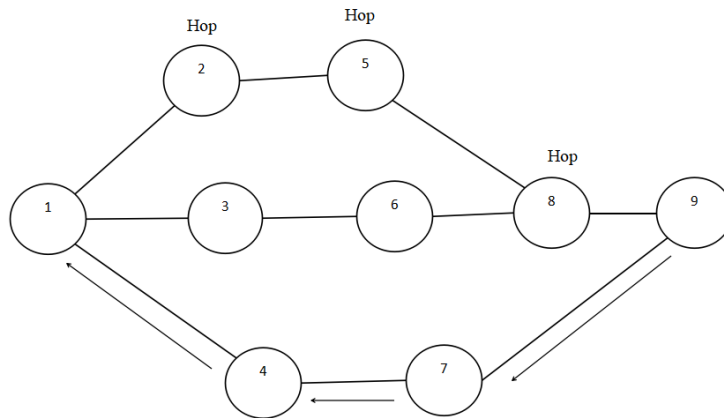


Figure 2.6: Route Reply from destination node 9 to source node 1.

*Temporarily Ordered Routing Algorithm(TORA):*

Each sensor node having the destination address makes it really simple as the node trying to send information only has to know the neighbor next to it. Here Directed Acyclic Graph (DAG) is followed by each node as each sensor node is independent of path creation rather than simply depending on the base station or sink node. Being on the demanding protocol it seems more beneficent, adjusted, sizable, multiple paths oriented, and contributive [9]. Figure 2.7, 2.8 shows query message and height update graph.

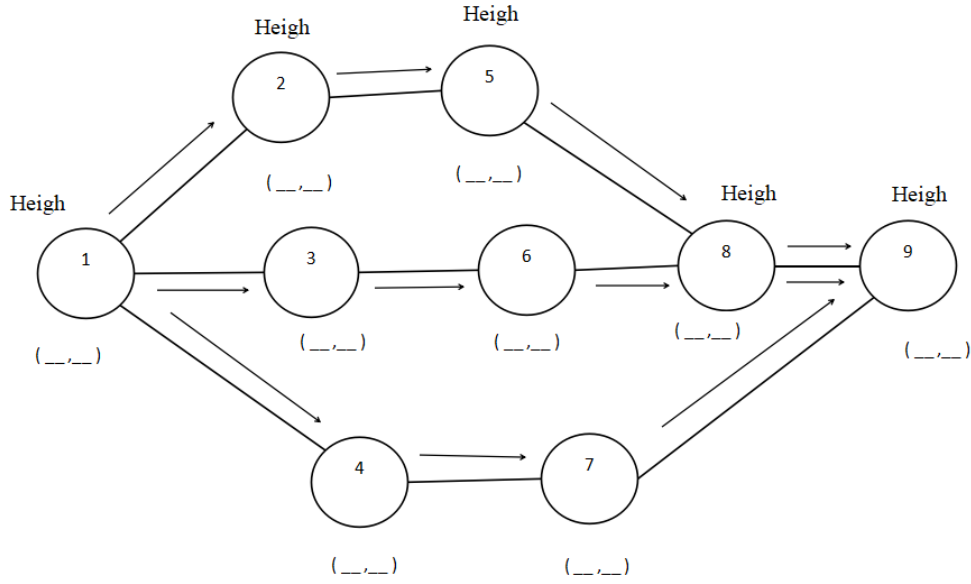


Figure 2.7: Propagation of Query message.

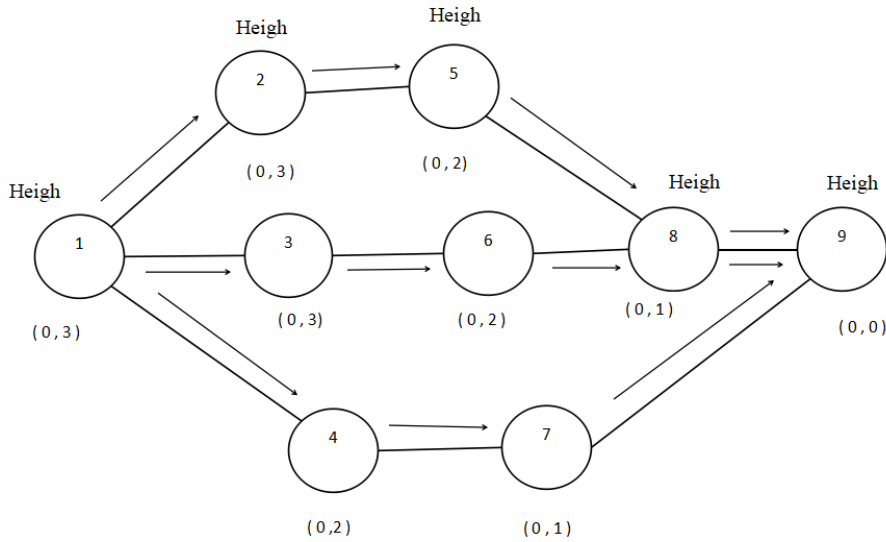


Figure 2.8: Node's height updated as a result of update message.

The algorithms discussed till now are all static routing protocols and LEACH performs by the dynamic routing protocol.

*Low-Energy Adaptive Clustering Hierarchy(LEACH):*

Among the algorithms that we have discussed so far, the LEACH is the most advanced of all. Low-Energy Adaptive Clustering Hierarchy a cluster-based protocol where it is a dynamic one. In this protocol, the cluster sink nodes perform a circular-based shift so that the energy load is evenly distributed. The algorithm of LEACH is divided into a couple of rounds where each round consists of preparing stages. This preparing stage gives a window to the next stage known as steady-state then

comes when the cluster head is organized. Only now and now data is being sent over the base station [10].

*i. Advertisement Stage:*

In this stage, nodes decide whether to become a cluster head node or just a typical node. With the help of prioritizing and percentage of cluster head in a particular topology and the number of times, the node has been cluster head so far it performs the decision-making process. The number is less than the assigned threshold  $T(n)$  the sensor node is the cluster head for the present round. Finally, this cluster node sends the broadcast information to the other nodes. Here CSMA MAC protocol is performed using the same transmission energy [10].

*ii. Cluster Set-Up Phase:*

Receiving the broadcast message from the cluster node then the sensor node residing the topology decides in which cluster head they belong to. However, for this phase, the cluster heads receivers are on [10].

*iii. Schedule Criteria:*

In this step a sensor node judges to be in a delicate cluster based on the set-up phase, the clutch leader conducts a TDMA program to the sensor nodes acknowledging that the sensor node can conduct data [10].

*iv. Data Transmission:*

Fix TDMA program initiates data movement. For this, an allocated transmission period is given. This is the steady-state regulation of LEACH networks. At the next round determined by prioritization, the next round proceeds and again the whole procedure repeats as mentioned above meaning a new clutch leader is selected [10].

*v. Hierarchical Clustering or Prioritization:*

The author proposed their work can be expanded to form hierarchical clustering. Each cluster head is compared with a super cluster node until the node is situated in the hierarchy table. It saves their ample amount of time and energy. They further mentioned that they will try to perform this without the help of base stations [10].

## Chapter 3

# WANLoc: Weight based Adaptive Neighbor Localization Algorithm

WANLoc is a self-adaptive localization algorithm that dramatically reduces the complexity of computations that SNs have to do to find candidates to transfer data to. It accomplishes this by reducing the number of neighboring nodes that SNs have to account for when it comes to deciding where to send data. WANLoc uses a highly optimized tables that can simply be looked up to and determine who to forward data packets to, not only making the network highly dynamic and only respond to changes made in the immediate neighborhood but also making them resistance to major network disruptions.

WANLoc configures the network through tables that are formed in specific phases for each node. At any given point in time nodes can be in two states: Dormant or Mature.

While in Dormant state, SNs simply advertise themselves by broadcasting information to each other in order to establish an intuitive understanding of nodes that are within close proximity. This is done by accumulating neighboring node information into a buffer that is sorted based on how close the neighboring nodes are. This list is further optimized by drawing up a Neighbor Table of significantly fewer SNs. At this point SNs simply keep optimizing the table until they discover the location of the SINK. This done by means of point to point information transfer or ‘gossip’. Once nodes receive the location of the SINK, they end their dormancy phase and begin assigning themselves a weight, based on how close they are to the SINK. Once weights have been assigned, further optimized tables are drawn up that separates the neighborhood into two categories based on weight. At this point the nodes begin their Linking phase where they establish minimum distanced connections with their immediate neighbors. This ensures that no overlapping connections occur, thus reducing interference greatly. It also ensures that packets are transferred over necessary minimum distances, thus reducing the probability of packet drops. Upon the completion of connection establishment, nodes mark themselves as Mature and begin their Data Transmission Cycle.

In WANLoc, once the entire network has achieved maturity, it becomes highly adaptive to change or any disturbance to the network. For example, the loss of a node or an addition of a node, only affects the connections in the immediate neighborhood only, allowing the network to stay virtually unchanged. This phenomenon makes WANLoc highly effective in dynamic networks, where nodes change their locations



i.e., they join or leave the network, while keeping computational complexity to a minimum by ensuring communication between only the directly connected neighboring nodes.

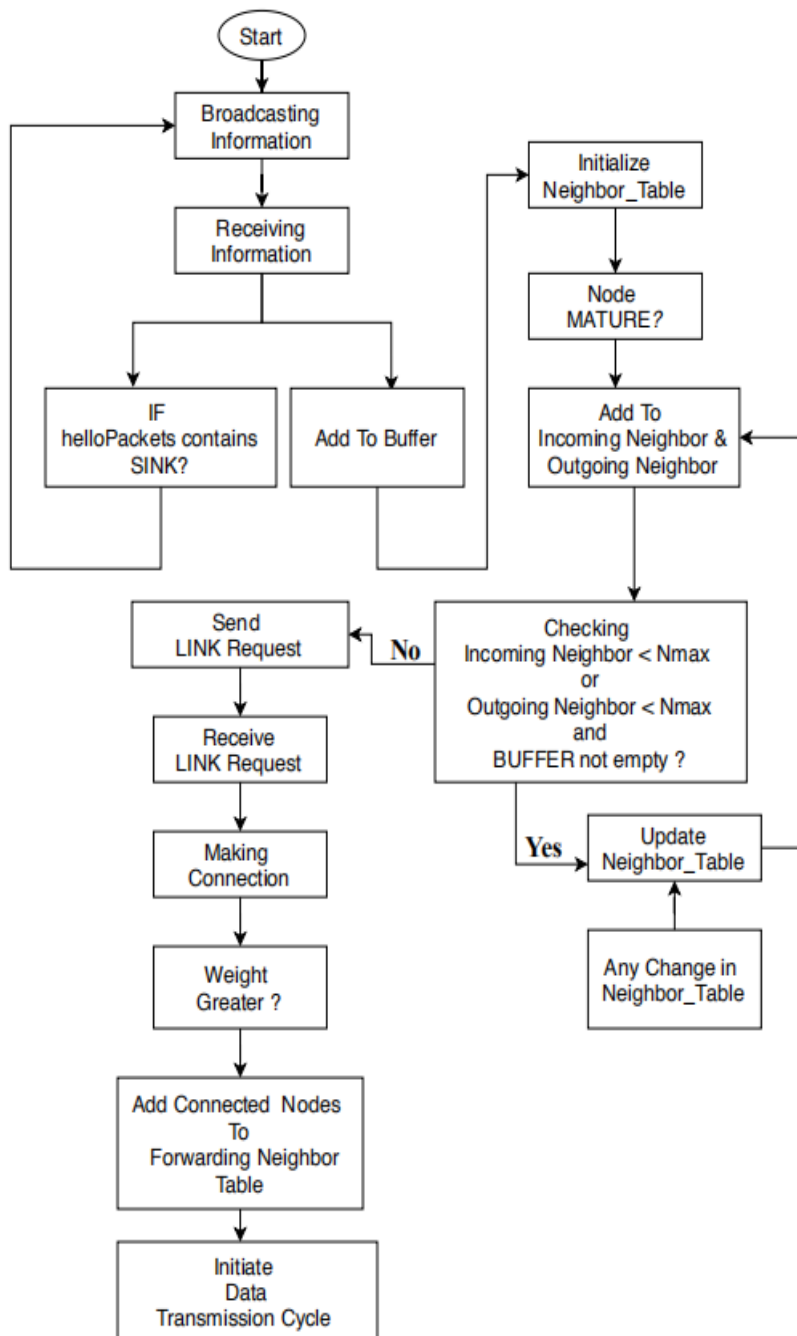


Figure 3.1: Flow Chart for WANLoc Algorithm.

## 3.1 Localization

To keep in line with our algorithm principles, data transfers between SNs must be quick and hence, carried out over as minimum distance as possible. This ensures that the probability of packet drops occurring remains a minimum.

In addition, we must also ensure that there is as less interference or cross connection data transfers in the network as possible. Therefore, data transfers must only occur between close neighboring SNs only.

If SNs are only interacting with their immediate neighbors, it is unnecessary for them to keep track of anything but those neighbors. SNs must maintain a list of said neighbors only, which will be considerably smaller and hence much quicker to traverse or lookup when selecting candidates for forwarding data. Therefore, an estimate for the size of this lists must be determined. We have found that the maximum number of close neighbors a SN can connect to without any cross-connection occurring is given by

$$N_{max,d} = d \times (d+1)$$

Where ‘d’ is the working dimension of the WSN i.e. 2D or 3D.

### 3.1.1 Geometric proof

Since WANLoc prioritizes minimum distanced links between SNs in( $r_{min}$ ) when forming connections, we will demonstrate the logic behind the  $N_{max}$  function by considering the worst-case scenario where every SN is equally distanced from each other (no  $r_{min}$ ).

Consider a sequence of points in a one-dimensional world (Figure 3.2). The maximum number of connections that can be drawn from point to point without any over laps is two.

$$N_{max,1} = 1 \times (1+1) = 2$$



Figure 3.2: Equidistant points in 1D.

Similarly, in a two-dimensional world (Figure 3.3), we end up with an equilateral triangle as the first primitive shape which is formed from three equally distanced points in 2D space. If we combined a number of these triangles from one common central point, we can draw a maximum of six. We get a regular hexagon which has six vertices excluding the center.

$$N_{max,2} = 2 \times (2+1) = 6$$

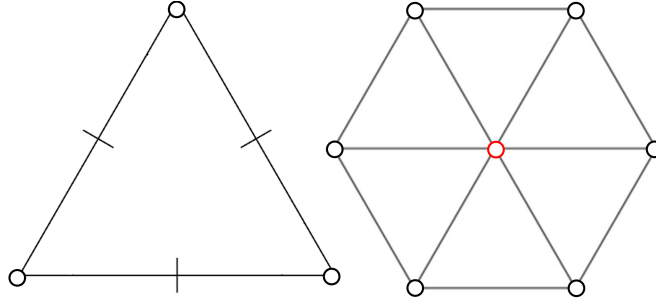


Figure 3.3: Equidistant points in 2D.

In a three-dimensional world (Figure 3.4), the first primitive shape is a regular tetrahedron which is formed from four equally distanced points in 3D space. Combining a number of if these together from a common point at the center, we get a regular icosahedron which has twelve vertices excluding the center

$$N_{max,3} = 3 \times (3+1) = 12$$

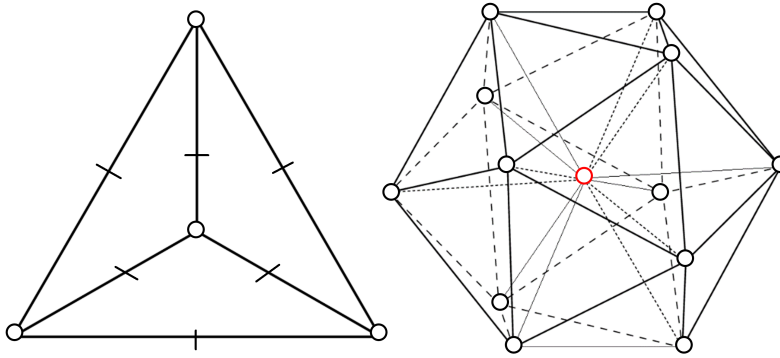


Figure 3.4: Equidistant points in 3D.

## 3.2 Weighted Nodes

We have previously established that WANLoc ensures each SNs only interact with their closest neighbors, i.e. all interactions are highly localized. However, this creates a major ambiguity during the Data Transfer Cycle. Since the SNs are not directly interacting with the SINK, SNs lack the sense of direction needed for them to set up a data transfer chain or pathway.

Thus, WANLoc uses a simple technique to control the flow of data; it assigns each node a weight(W).

The concept of weight works the same way as it does in the physical world. Since the earth is considered the source of gravity every object close to it is attracted to it. Objects, when closer to the earth have a higher weight compared to when they are further. The weight function depends on the distance(R) from the earth and has an inverse relationship to it. Similarly, WANLoc assigns normalized weight(W) to each SN depending on its location compared to the SINK. This allows the SNs to transfer data only towards the ‘center’ just by sending it to the ‘heaviest’ neighbor it is connected to.

### 3.2.1 Function Comparison

The weight function depends on the distance between the SN and the Sink ( $R$ ). This is essentially the Euclidean distance between the SN and the SINK and hence, can be calculated using the two nodes' location information (i.e. their coordinates).

$$R = \sqrt{((x_0 - x_s)^2 + (y_0 - y_s)^2 + (z_0 - z_s)^2)}$$

where  $(x_0, y_0, z_0)$  is the location of the SN and  $(x_s, y_s, z_s)$  is the location of the nearest SINK

We can then use this value to calculate  $W$ . When determining the weight function, we looked at two different mathematical equations:

$$W_1 = \frac{1}{(R+1)}, R \geq 0$$

$$W_2 = \frac{1}{1+\log(R+1)}, R \geq 0$$

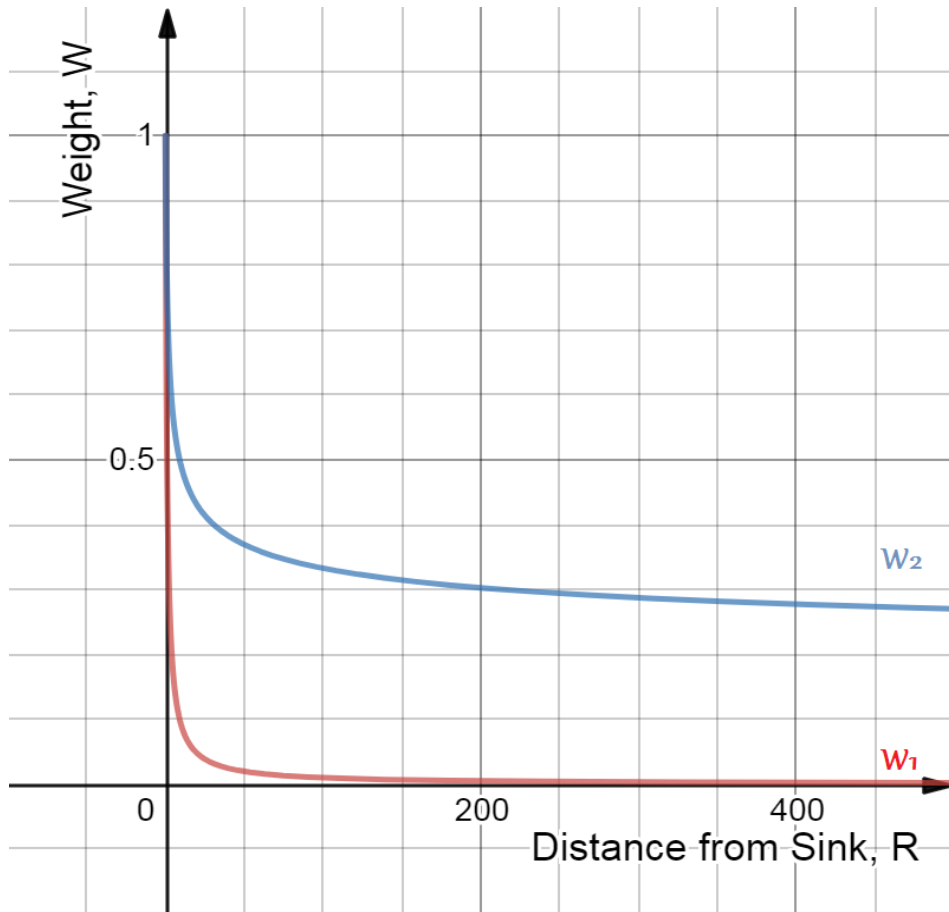


Figure 3.5: Weight vs. Global Radius.

Looking at the graphs of the two functions (Figure 3.5), we can see that both the functions gives us a normalized value. The general function  $W_1$  produced a very narrow graph that radically reduces the value of the weight over a relatively small change in  $R$ . This would constrain the network size greatly since the  $W$  will become too small and virtually non-existent for SNs far away from the SINK. However, the

logarithmic function,  $W_2$  gives us a steadier  $W$  over larger values of  $R$ . We chose this function for determining weight in WANLoc since it will make the network highly scalable.

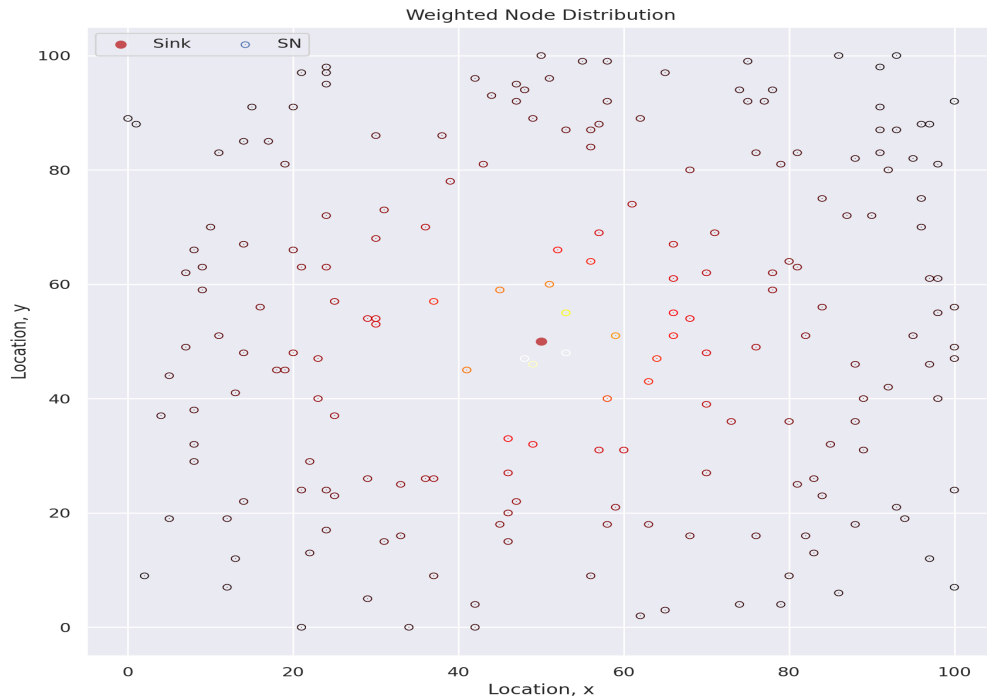


Figure 3.6: Weight assignment in WANLoc.

In the randomly generated node population illustrated in Figure 3.6 We can clearly see the effect of  $W$  in the heatmaps. As nodes get closer and closer to the SINK, their weight increases. This provides a simple yet highly effective means of identifying the possible location of the SINK. Further intuitive visualization of the data can give us an even clearer picture as shown in the 3D heatmap illustrated in Figure 3.7 and Figure 3.8.

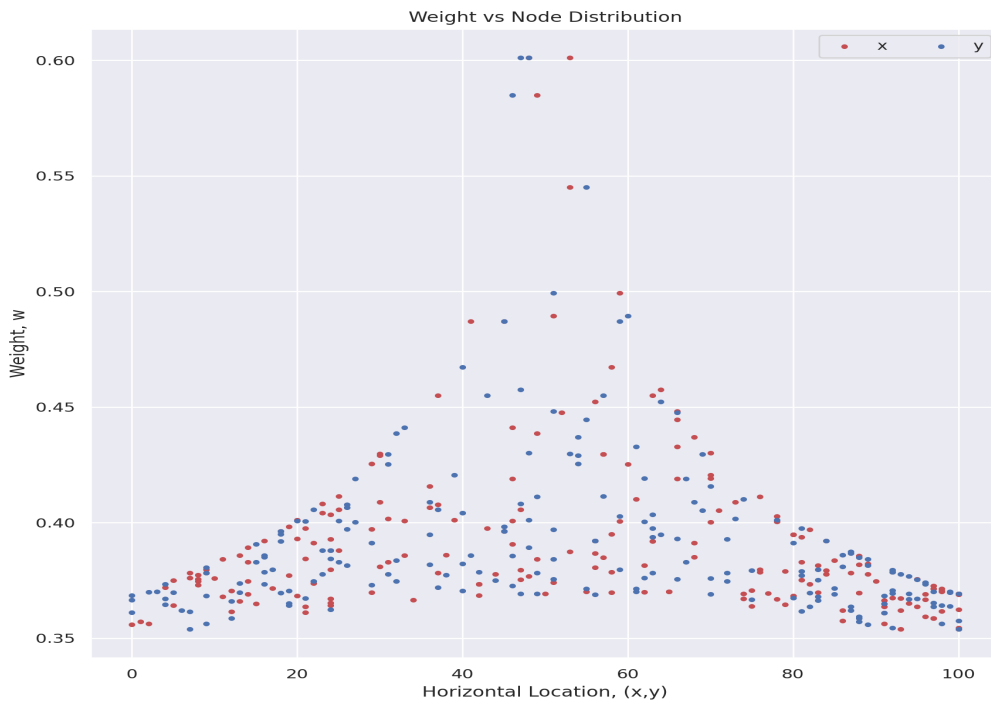


Figure 3.7: Weighted SNs

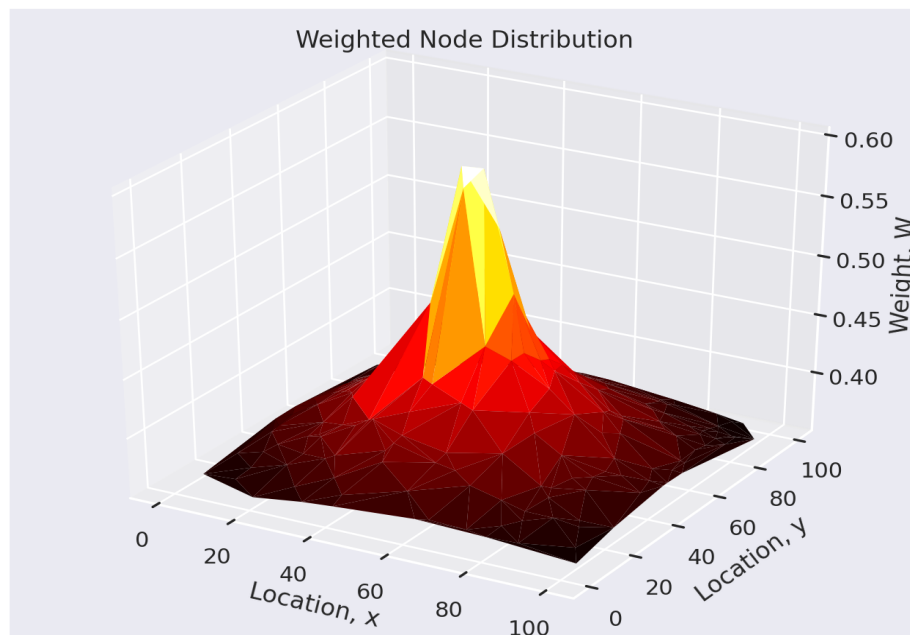


Figure 3.8: Heatmap based on Weight

WANLoc also supports multi-sink networks as well. In this case, nodes choose the closer SINK when calculating their weights. This effect is clearly illustrated in Figure 3.9 and Figure 3.10.

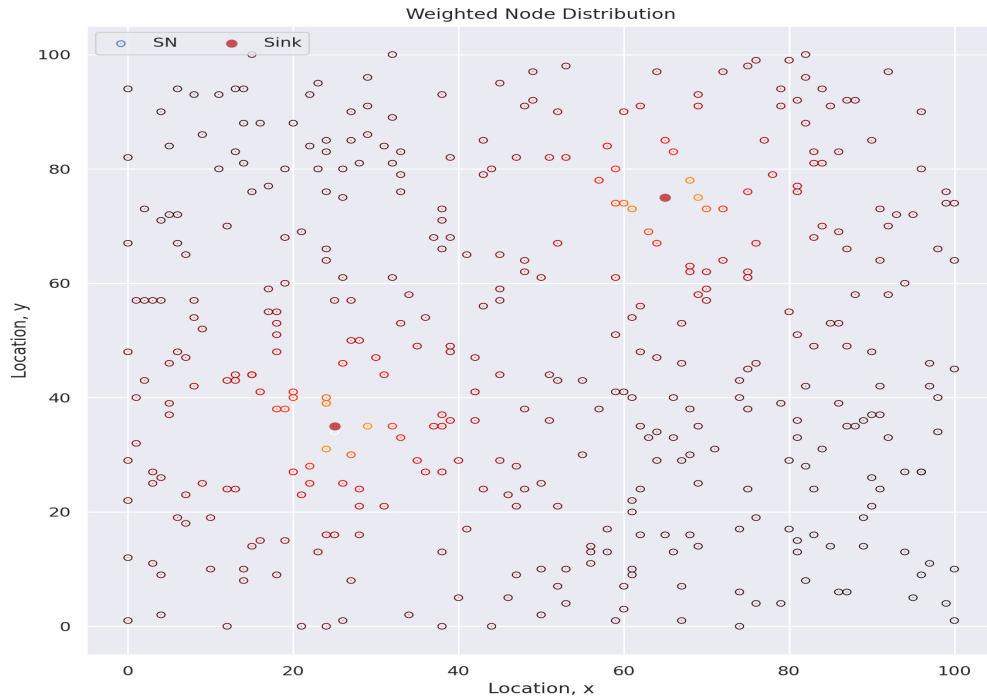


Figure 3.9: Weight assignment with multiple SINKs

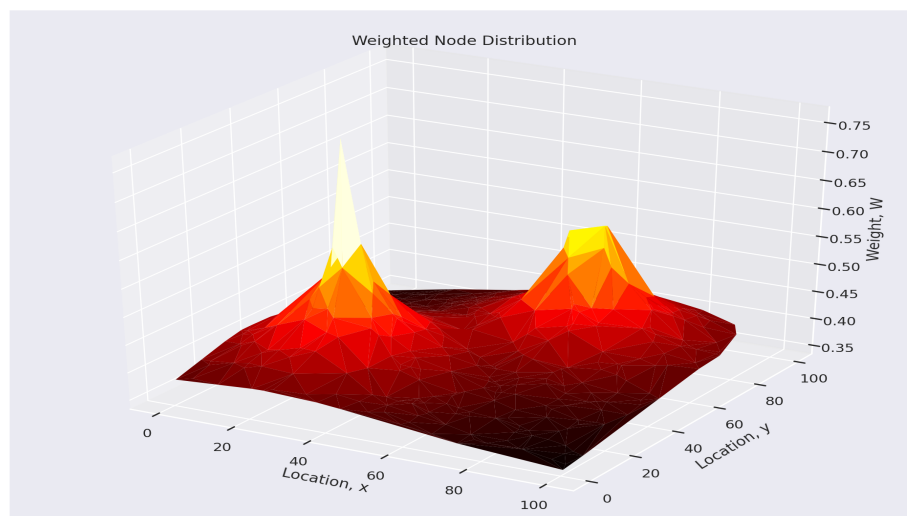


Figure 3.10: Heatmap of a Multi-SINK network

# Chapter 4

## Implementation and Result

Here, we will explore the inner mechanism of the entire WANLoc Algorithm while in action. The steps and methods are sequentially summarized below. Since we are exclusively working on just the algorithm while keeping every other protocol associated to data transmissions between devices constant and abstracted, we decided to design, implement and simulate the entire model on Python 3. The simple pseudo code like structure of the Python language makes it an ideal candidate for designing and testing algorithms.

All SNs begin their activity in the dormant phase and eventually reach maturity after which they can begin and continue to Data Transmission without further computations.

### 4.1 WANLoc Parameters

In order to make a network that is both robust and dynamic, an efficient network creation phase is a necessity. In order to reach maturity in WANLoc, we have determined the basic parameters nodes need to have there are summarized in the table below.



Parameters	Description	Type, Size
ID	unique identifier	string/int,n/a
Loc	location or coordinates of the node (x, y, z)	int/float,3
W	weight of the node relative to the nearest sink	float,1
d	working dimension of the network (2D or 3D)	int,1
Mature	denotes node phase	Boolean,n/a
Dormant	denotes node phase	Boolean,n/a
buffer	Space for hello packets	2D list,depends on range
sink	sink locations	2D list,no.of sinks
-	<b>Tables</b>	-
$n_{table}$	comprehensive neighborhood table	2D list, $0 < x \leq 2N_{max}$
$n_{in}$	list of nodes data will be received from	list, $0 \leq x \leq N_{max}$
$n_{out}$	list of nodes data can be forwarded to	list, $0 < x \leq N_{max}$
$w_{out}$	active candidates for data transmission	2D list, $0 < x \leq N_{max}$

Table 4.1: SN parameters in WANLoc

WANLoc Algorithm is comprised of a number of auxiliary algorithms that run sequentially to construct the highly optimized tables, between two states. These states and their algorithms are summarized below.

---

**Algorithm 1:** Calculating Weight of SNs

---

*calcR()* = function to calculate distance between two nodes  
**Input:** *loc, sink*  
**Result:** *W*  
**begin**  
     $R \leftarrow [0] \times \text{length}(\text{sink});$   
    **for** *each member N in sink* **do**  
         $R \leftarrow \text{append } \text{calcR}(\text{loc}, \text{sink});$   
    **end**  
    sort *R*;  
     $W \leftarrow 1/(1 + \log(R[0] + 1));$   
    **return** *W*;  
**end**

---

## 4.2 Dormant State

This is the initial state of the network where all the nodes have just been initialized. The following series of Algorithms are executed. Algorithm 2 prepares the hello packets that are to be broadcast in the form:

$$[ \text{ID}, \text{loc}[ \text{x}, \text{y}, \text{z} ], \text{W}, \text{sink}[] ]$$

---

**Algorithm 2:** Prepare Broadcast Information Packets

---

**Input:** *ID, loc, W, sink*  
**Result:** *helloPacket*  
**begin**  
    *helloPacket*  $\leftarrow$  append *ID*;  
    *helloPacket*  $\leftarrow$  append *loc*;  
    *helloPacket*  $\leftarrow$  append *W*;  
    *helloPacket*  $\leftarrow$  append *sink*;  
    **return** *helloPacket*;  
**end**

---

Once the helloPackets are prepared these are broadcasted to all other SNs in range. They are then received and processed by each SN independently. First, they are modified with a few additional information and are added to the buffer in the form:

$$[ \text{ID}, \text{loc}[ \text{x}, \text{y}, \text{z} ], \text{W}, \text{r}, \text{linkRequested}, \text{linkConnected} ]$$

This process is summarized in Algorithm 3 shown below.

---

**Algorithm 3:** Receiving broadcast-ed information and adding it to Buffer

---

*calcR()* = function to calculate distance between two nodes  
*calcW()* = function to calculate weight of a node (Algorithm 1)  
**Input:** *helloPacket, loc, sink, Dormant*  
**Result:** *buffer*  
**begin**  
    *info*  $\leftarrow$  append *helloPacket*[3];  
    *info*  $\leftarrow$  append *calcR*(*loc, sink*);  
    *info*  $\leftarrow$  append *False*;  
    *info*  $\leftarrow$  append *False*;  
    **if** *helloPacket*[2]  $\neq \emptyset$  **then**  
        *sink*  $\leftarrow$  append *helloPacket*[3];  
        *W*  $\leftarrow$  *calcW*(*loc, sink*);  
        *info*[2]  $\leftarrow$  *calcW*(*info*[1], *sink*);  
        *M*[2]  $\leftarrow$  *calcW*(*M*[1], *sink*) for all members *M* in *buffer*;  
        *Dormant*  $\leftarrow$  *False*;  
    **end**  
    *buffer*  $\leftarrow$  append *info*;  
    sort *buffer* based on *r* value of its members;  
    **return** *buffer*;  
**end**

---

Once the buffer is prepared, the Neighbor Table is drawn up using Algorithm 4 displayed below.

---

**Algorithm 4:** Drawing up the initial Neighbor Table

---

**Input:** *buffer, N<sub>max</sub>*  
**Result:** *n\_table*  
**begin**  
    **for** each member *N* in *buffer* **do**  
        **if** length of *n\_table*  $< 2 \times N_{max}$  **then**  
            *n\_table*  $\leftarrow$  append *N*;  
        **else**  
            **break**  
        **end**  
    **end**  
    **return** *n\_table*;  
**end**

---

Recall that Algorithm 3 sets Dormant to False when a SINK is discovered. This triggers the continuation to the algorithms stated in the Mature State. Otherwise, SNs simply loop through Algorithm 3 and Algorithm 4.

### 4.3 Mature State

Now the SN is ready for Data Transmission. But before that the node has to establish minimum distanced connections with other SNs it will interact with. These are the closest neighbors with whom non-overlapping links can be created. This is accomplished by executing the following algorithms.

First, the Weight based Incoming and Outgoing Tables are drawn.

---

**Algorithm 5:** Adding and Updating the Incoming and Outgoing Neighbor Tables

---

*popBuffer(N)* removes the element  $N$  from *buffer*

**Input:**  $n\_table, W, buffer, N_{max}$

**Result:**  $n\_in, n\_out$

**begin**

$n\_in \leftarrow$  empty set;

$n\_out \leftarrow$  empty set;

**while**  $buffer \neq \emptyset$  **do**

**if**  $length\ of\ n\_in < N_{max}$  **or**  $length\ of\ n\_out < N_{max}$  **then**

            run **Algorithm 6**;

**for** each member  $N$  in  $n\_table$  **do**

**if**  $N$  not in  $n\_in$  **or**  $N$  not in  $n\_out$  **then**

**if**  $N[2] > W$  **then**

**if**  $length(n\_out) < N_{max}$  **then**

$n\_out \leftarrow$  append  $N$ ;

*popBuffer(N)*;

**end**

**else**

**if**  $length(n\_in) < N_{max}$  **then**

$n\_in \leftarrow$  append  $N$ ;

*popBuffer(N)*;

**end**

**end**

**end**

**end**

**end**

$buffer \leftarrow$  empty set;

**end**

    sort  $n\_in, n\_out$  based on the  $r$  value of their members;

**return**  $n\_in, n\_out$ ;

**end**

---

Algorithm 5 loops through in conjunction with Algorithm 6 in order to iterate through every entry in both the Neighbor Table and the buffer while dropping redundant information from each. These are the initial Optimization Steps that ultimately generates the minimized tables that can allow the SNs to form small neighborhoods that reduces the computational complexity when deciding on Data Transmission candidates.

---

**Algorithm 6:** Updating the Neighbor Table and Buffer

---

*popNtable(N)* removes the element  $N$  from  $n\_table$

**Input:**  $n\_table, n\_in, n\_out, buffer, N_{max}$

**Result:**  $n\_table, buffer$  (Updated)

```
begin
  for each member  $N$  in  $n\_table$  (reverse) do
    if  $N$  not in  $n\_in$  and  $N$  not in  $n\_out$  then
      entry  $\leftarrow$  empty set;
      for each member  $M$  in  $buffer$  do
        if  $M[3] > N[3]$  then
          entry  $\leftarrow M$ ;
          break ;
        end
      end
      if entry  $\neq \emptyset$  then
         $N \leftarrow$  entry;
        break ;
      else
         $buffer \leftarrow$  empty set;
        popNtable(N);
      end
    end
  end
end
```

---

Once the two tables have been formed and the buffer cleared, SNs begin sending the top  $N_{max}$  members of their respective Neighbor Tables ‘link requests’. SNs that receive these requests sets the requester’s linkRequested (see Algorithm 3) value in their respective tables as True. SNs then set the top  $N_{max}$  requester’s linkConnected (see Algorithm 3) value to True. These two simple steps allow the SNs to dynamically establish non-overlapping connections with their immediate neighbors, furthers reducing computations needed.

A final sorted Forwarding Table is generated by executing Algorithm 7 that holds the heaviest connected output neighbors on top of the list, thus allowing SNs to decide candidates through a simple lookup.

---

**Algorithm 7:** Drawing up the Forwarding Table

---

**Input:**  $n\_out, N_{max}$ **Result:**  $w\_out$ 

```
begin
  for each member  $N$  in  $n\_out$  do
    if  $N$  is a connected Neighbour then
      |  $w\_out \leftarrow$  append  $N$ ;
    end
  end
  sort  $w\_out$  based on  $W$  value of its members;
  return  $w\_out$ ;
end
```

---

To sum it up, the total algorithm for WANLoc can be denoted by the following algorithm.

---

**Algorithm 8:** WANLoc Algorithm

---

**Input:**  $ID, loc, W, sink, buffer, N_{max}, helloPackets, Dormant, Mature$ **Result:**  $w\_out, n\_in, n\_out, n\_table$ 

```
begin
  while  $Dormant$  do
    | run Algorithm 2;
    | run Algorithm 3;
    | run Algorithm 4;
  end
  while  $buffer \neq \emptyset$  do
    | run Algorithm 5;
  end
  run Algorithm 7;
  initiate Data Transmission;
  return  $w\_out, n\_in, n\_out, n\_table$ ;
end
```

---

## 4.4 Result Analysis

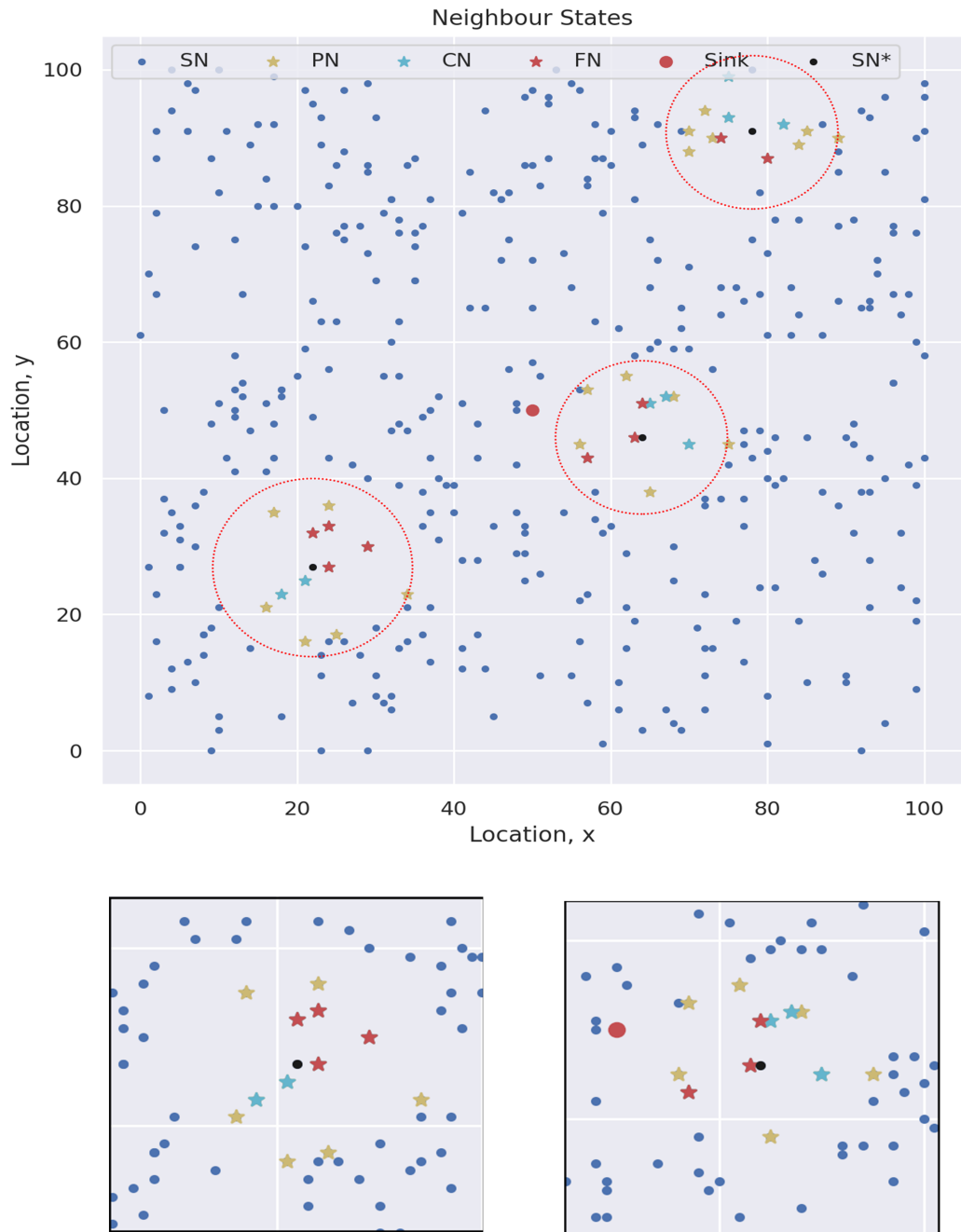


Figure 4.1: WANLoc simulation result on 400 SN network.

Figure 4.1 above illustrates a resulting network created by running WANLoc Algorithm. This network was generated with 400 SNs distributed randomly over a 100x100 grid with the SINK node placed at the location (50,50).

After the formation of the network through WANLoc, we are taking a glimpse at three randomly picked SNs (marked as SN\*) and the members of their respective Neighbor Tables. Here, we can see the positions of the Potential Neighbors (PN)

which includes the Connected Neighbors (CN), which in turn includes the Forwarding Neighbors (FN).

As predicted, the data above shows that WANLoc is successfully creating the highly optimized tables for each SN that it can use to make simple data transmission decisions without having to recursively look for pathways. The presence of the small neighborhoods (mini clusters) also proves our working principle that nodes can make heuristic predictions about the location of the SINK without ever having to interact with it; thus, making them network independent without having a star topology.

The graph clearly shows that the Forwarding Nodes are always the neighbors closer to the SINK. It also shows that the connected nodes are always the neighbors closest to the SN and never exceed  $N_{max}$ . Figure 4.2 shows the simulation of WANLoc on a sparse WSN of 200 SNs.

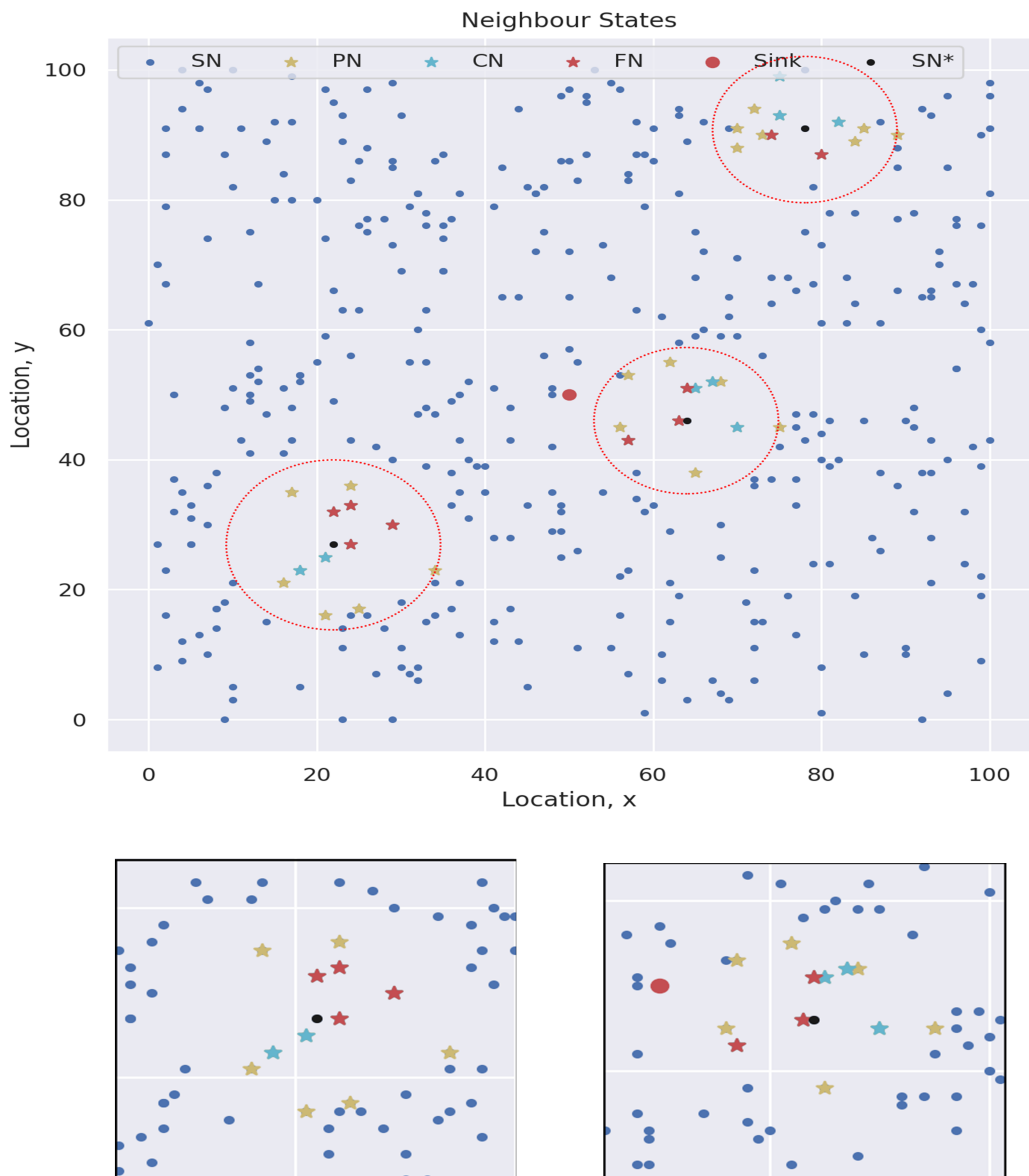


Figure 4.2: WANLoc simulation result on 200 SN network.



## 4.5 Advantages

Taking into account all the points regarding WANLoc explained above and the simulation results, we can claim some distinguishing characteristics of the algorithm that can allow it to stand out. We will be summarizing them in terms of two following aspects:

### 4.5.1 Complexity

As we have discussed above WANLoc was inspired by existing mesh and hierarchical algorithms where we have tried to bring the best of both worlds together while covering each other's weaknesses. Thus, our algorithm is superior to any of the existing algorithms on its own.

The mesh algorithms mentioned in this paper still have their computational complexity quite high and expect little to no disturbance to the network. Thus, they remain unsuitable for dynamic WSNs which would require continuous network reconfiguring and therefore heavy computational loads. This will also increase latency and data loss in the network.

On the other hand, LEACH is one of the best hierarchical WSN algorithms that is highly dynamic and energy efficient and can therefore be expected to fare well against small disturbances to the network. However, in LEACH, the hierarchy or clusters are reconfigured after every Data Transmission cycle. This means that the WSN is transferring data only half the time while the other half is spent computing clusters and electing cluster heads.

WANLoc however only configures the network once at the beginning. As SNs achieve maturity they begin transmitting data immediately, irrespective of the state of the network as a whole (i.e. the process is network independent). And once the entire network reaches maturity, there is no further need for any kind of re-configuring unless there are any disruptions to the network. Even in the event of a disruption only the affected neighborhood must undergo reconfiguration. As illustrated above, these neighborhoods are highly localized and minimized. Thus, any affect will be virtually invisible to the network as a whole.

Perhaps the greatest edge WANLoc has over any existing algorithms is that it not only builds up the network in a highly dynamic fashion but also gives the network a unique architecture which can well accommodate change. To summarize, SNs in WANLoc spend only one initial cycle configuring their connections after which they can simply focus transferring data virtually uninterrupted, greatly reducing overall network and computational complexity, and conserving a lot of energy in the process as well.

## 4.5.2 Scalability

As illustrated in Figure 10, SNs in WANLoc only need to communicate with their immediate neighboring nodes to function effectively. They don't need to have a direct link to the SINK at any point in time. This means the sensor network is no longer constrained by any effective range (usually the maximum range of the SINK, e.g. in LEACH). WANLoc will integrate any number of additional SNs added at the fringes the same way it does around the center of the network, making the network highly scalable.

In addition, SNs can now be designed with reduced broadcasting range and power since will only be communicating over minimum distances. This means SNs will have more conserved energy available as well. This fact also highlights a key property of our algorithm, the more concentrated and larger the network is the better WANLoc performs.

Another major advantage WANLoc has over any other algorithm is that is has been to account for SNs placed of 3D space. Switching from 2D to 3D is as effortless as changing the value of a variable. Thus, networks implementing WANLoc can not only be scaled outwards over a plane but also a volume of space.

<b>Scenarios</b>	<b>Mesh Algorithm</b>	<b>Hierarchical Algorithm</b>	<b>Proposed Algorithm</b>
Recursive Lookup	Yes	Yes	No
Heuristic Predictions to the sink node	Yes	No	Yes
Network Dependency	Yes	Yes	No
Communicating with sink node on topology establishment	Yes	Yes	No
Handling Dynamic Network	Not Good	Partially Good	Better than Hierarchical algorithm
Computational Complexity	High	Average	One time. On the creation of the topology
Latency and Data loss	High	Low	Low
Network Disruption	Complete topology re-establishment	Complete topology re-establishment	Only affected locality re-configures
Network Size	Restricted	Restricted	Unrestricted
Load Balancing	Bad	Better	Bad
Energy consumption	High	Low	Lower than Mesh

Table 4.2: Logical Comparison of Algorithms

# Chapter 5

## Conclusion

This paper proposed a simple weight based adaptive neighbor localization algorithm (WANLoc) to reduce the complexity of creating a dynamic mesh topology that can work in a dynamic WSN in a highly efficient way. Its unique architecture can easily accommodate any changes to the network. Only in the beginning, it configures the network which does not need any further reconfiguration unless there is any disruption in the network in which case the effect of the disruption is kept confined a small localized neighborhood, keeping the network as a whole virtually uninterrupted. As we have demonstrated above, preliminary iterations of the algorithm show that it can effectively configure a dynamic network without looking at the entire network as a whole. We hope to further optimize the codes in the algorithm to create even more tighter neighborhoods resulting in increased resistance to major network failure.

### 5.1 *Application*

As we mentioned above, in the near future we expect to see a massive increase in the use of SNs including its integration into drones. Possible scenarios can include industrial usage in natural resource prospecting in regions where humans cannot access but drones can. Also note that WANLoc supports 3D application of WSNs meaning it can also be used in situations where SNs have to work not only in a single flat plane but over a certain volume of space. For example, fire fighters can deploy drones with SNs inside collapsed structures to look for signs of life, where the network transmission range is minimums due to rubble. WANLoc can be an excellent choice in such cases.

# Bibliography

- [1] C. Hu, W. Bao, and D. Wang, "IoT Communication Sharing: Scenarios, Algorithms, and Implementation," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 1556-1564, DOI: 10.1109/INFOCOM.2018.8486329.
- [2] S. H. Park, S. Cho, and J. R. Lee. (2014). "Energy-Efficient Probabilistic Routing Algorithm for Internet of Things". Journal of Applied Mathematics, vol. 2014, Article ID 213106. DOI:10.1155/ 2014/213106.
- [3] Chhabra, A., Vashishth, V., Khanna, A., Sharma, D.K., and Singh, J., 2018. "An energy-efficient routing protocol for wireless internet-of-things sensor networks". arXivpreprintarXiv:1808.01039.
- [4] Zhang, J., Yang, T. and Zhao, C. (2016) 'Energy-efficient and self-adaptive routing algorithm based on event-driven in wireless sensor network', Int. J. Grid and Utility Computing, Vol. 7, No. 1, pp.41-49.
- [5] D. Evans, "The Internet of things: How the next evolution of the Internet is changing everything". Cisco IBSG, San Francisco, CA, USA, April 2011. [Online]. Available:<http://www.cisco.com/web/about/ac79/docs/innov/IoT-IBSG-0411FINAL.pdf>
- [6] The Zettabyte Era-Trends and Analysis. Cisco, May 2013.[Online]. Available:<http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI-Hyperconnectivity-WP.html>
- [7] Maayan. G. D. (2020). "The IoT Rundown For 2020: Stats, Risks, and Solutions". Available at: <https://securitytoday.com>
- [8] Liu, Y. and Tong, K., 2017. Wireless Mesh Networks In IoT Networks - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/abstract/document/7968828>
- [9] Vijayakumar, K., Ganeshkumar, P., and Anandaraj, M., 2012. Review On Routing Algorithms In Wireless Mesh Networks. [online] Ijcst.org. Available at: <http://www.ijcst.org/Volume3/Issue5/p15-3-5.pdf> .
- [10] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H., 2000. Energy-Efficient Communication Protocol For Wireless Microsensor Networks - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/abstract/document/926982> .

- [11] O. Bello and S. Zeadally, "Intelligent Device-to-Device Communication in the Internet of Things," in *IEEE Systems Journal*, vol. 10, no. 3, pp. 1172-1182, Sept. 2016, DOI: 10.1109/JSYST.2014.22 98837.
- [12] Tao F, Wang Y, Zuo Y, Yang H, Zhang M. Internet of Things in product life-cycle energy management. *Journal of Industrial Information Integration* 2016; 1: 26–39.