Shortest Path Computation Technique in WSN for IOT Applications

Ramandeep Gill (raman89gill@gmail.com)
Manipal University - Jaipur Campus https://orcid.org/0000-0003-0162-0159
Tarun Kumar Dubey
Manipal University - Jaipur Campus

Research Article

Keywords: Wireless Sensor Network, Internet of Things, Smart mobility, Sensor node

Posted Date: February 25th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1030832/v1

Shortest Path Computation Technique in WSN for IOT Applications

Ramandeep (RG) Gill[1], Dr. Tarun (TKD) Kumar Dubey[2]

[1]Department of Electronics and Communication Engineering, JECRC University, Sitapura, Jaipur, Rajasthan 303905, India

[2]Department of Electronics and Communication Engineering, Manipal University Jaipur, Dahmi Kalan, Jaipur, Rajasthan 303007, India

ABSTRACT

A Wireless Sensor Network (WSN) consists of a large no of sensor nodes deployed inside a physical environment that are capable of sensing and processing the data related to that environment and communicating it over the network. One of the vital applications of WSN is its suitability to smart cities that consist of many features such as smart buildings, smart education, smart governance, smart security, and smart mobility. This paper focuses on calculating the shortest path from the source node to the destination node in WSN by means of Dijkstra's Algorithm. Various other algorithms are also compared and their suitability to different application scenarios is discussed. Simulations have been made for the smart city and Internet of Things (IoT) application by using the WSN simulator CupCarbon U-One 4.2.

KEYWORDS: Wireless Sensor Network, Internet of Things, Smart mobility, Sensor node.

INTRODUCTION

WSN includes spatially distributed independent devices using sensors to cooperatively keep track of physical or environmental conditions, such as temperature level, audio, vibration, stress, motion, or toxins, at various locations [1]. It consists of a large number of sensor nodes that are randomly deployed in an environment. A basic sensor node comprises six main components: Power supply, Transceiver, Micro-Controller, External Memory,

and Sensors or Actuators.

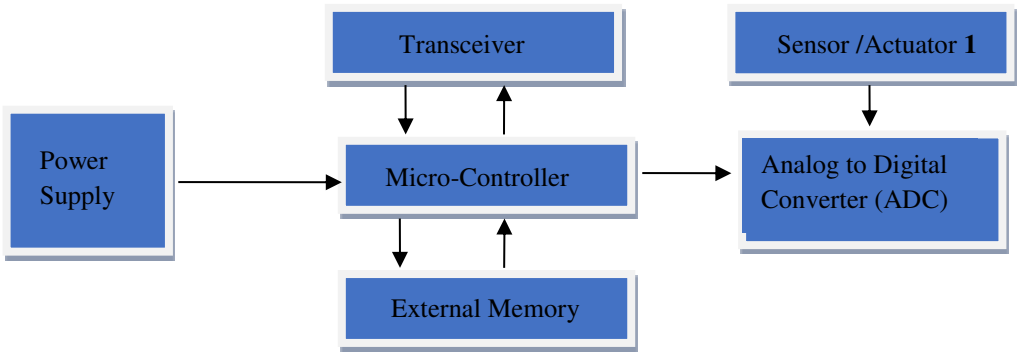

Fig 1. Components of a sensor node

The sensor nodes can sense and collect data, convert the data in Digital/Analog form using Analog to Digital Converter (ADC), process the data in Micro-Controller and transmit over a communication network using transceiver to the neighbouring nodes or the gateway which is connected to some user. As a result of current advancements in sensor technology and low power consuming radio frequency design, WSNs have gained appeal. WSNs are powerful as well as support a lot of real-world applications in the field of health, military, disaster relief, agriculture, smart cities, etc.

In metropolitan locations, the combination of the IoT and "sensing as a service" ideas with traditional WSN based systems is leading to the change of traditional city services in the direction of smart cities. This paper is focused on one of the many features of a smart city i.e., smart mobility [2] that finds applications in many areas such as smart exit, smart traffic management, smart accident management, and smart van for use in a hazardous environment. The key parameter in all these applications is to find the shortest route possible among all the routes to reach the destination [3]. In WSN there are various constraints and challenges such as limited battery life, small memory for processing and storage of data, small bandwidth for communication, Quality of Service, reliability, and packet loss [4]. Calculating the shortest route to the destination will certainly attend to a few of the constraints mentioned above and will result in much less usage of battery, reduction in the redundancy of data, less usage of the number of sensor nodes, and less usage of the bandwidth for communication.

RELATED WORK

I. Divide and Conquer Technique

In the divide and conquer technique a large network is divided into smaller sub-networks, required data is collected from these sub-networks and all this information is combined at last to get the desired output [5]. The basis of this technique is shown in Fig 2.

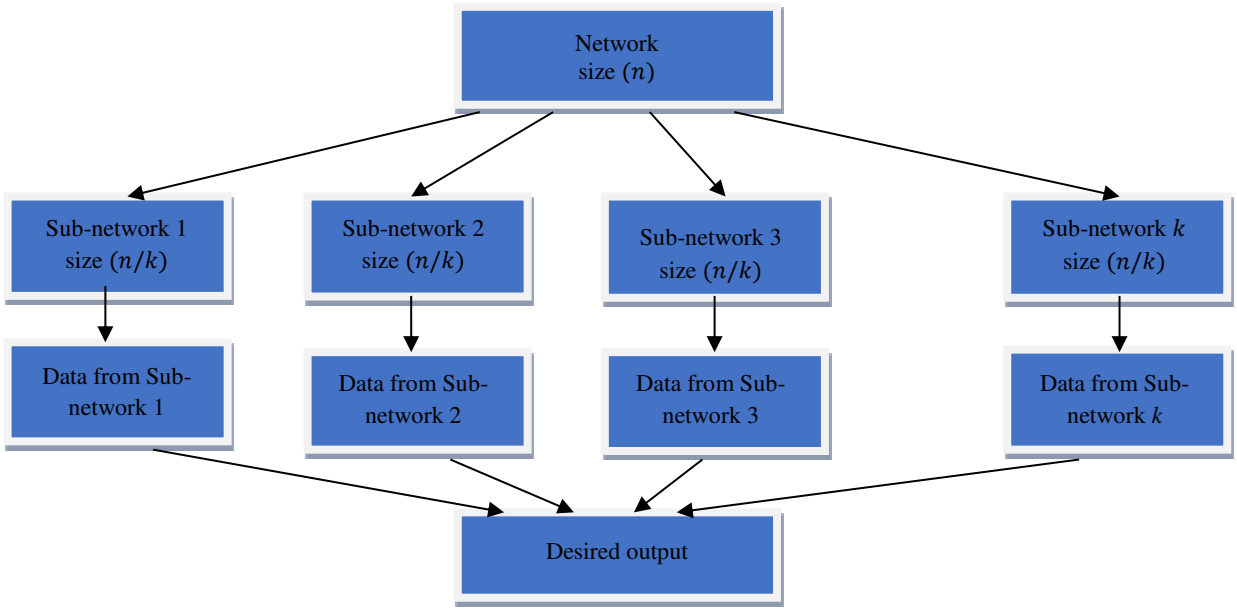

Fig 2. Divide and Conquer technique

In this technique, recursion is used at every step i.e., large network is divided into smaller sub-networks recursively, if sub-networks are small then data can be collected directly otherwise, they are divided further recursively until no other division is possible. Recursion is used to collect and combine the data to get the required output.

II. Bellman Ford's Algorithm

Bellman-Ford's algorithm is a single source shortest path algorithm i.e., in a directed weighted network, one node is selected as the source node and the shortest path from the source node to all other nodes is calculated [6]. The weights of the network can be negative numbers. It follows a dynamic programming strategy in which all the possible solutions are tested and the best solution is selected [6]. In this algorithm, all the edges of the network are relaxed $n-1$ times where n is the number of nodes and the distance of each edge is calculated in every repetition [6].

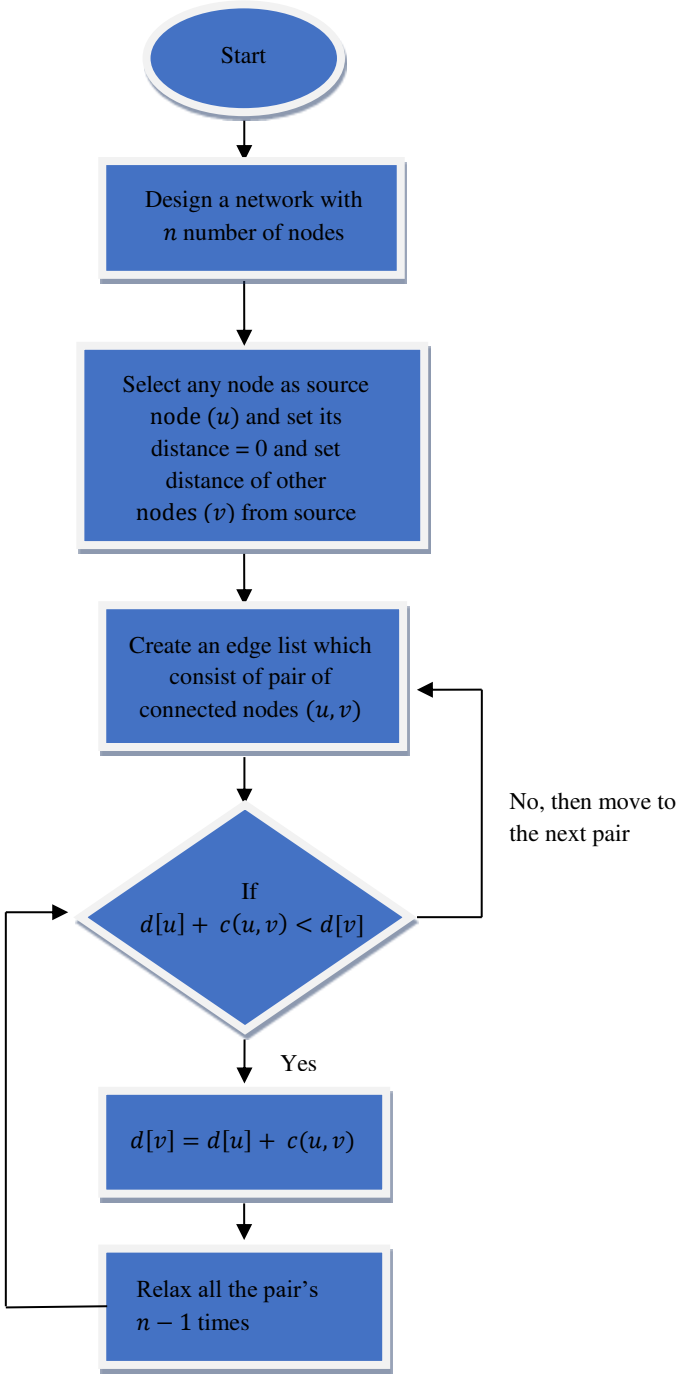

Fig 3. Flow Chart of Bellman Ford's Algorithm

III. Dijkstra's Algorithm

Dijkstra's algorithm is also a Single Source Shortest path algorithm and follows the Greedy method in which the problem is solved in stages considering one input at a time [7]. There is a predefined procedure to get an optimal solution which is described with the help of the following flow char

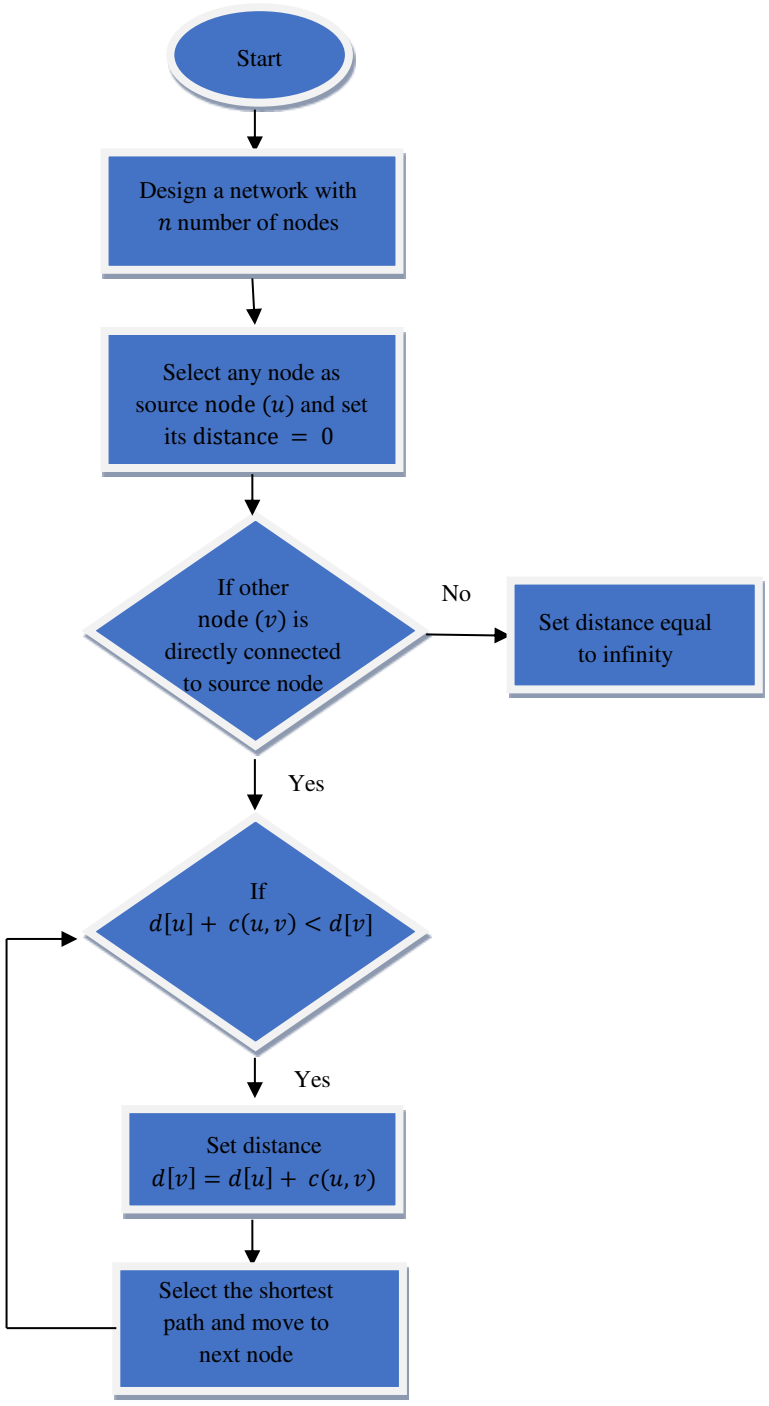

Fig 4. Flow Chart of Dijkstra's algorithm

It does not work for networks with negative weights which is not an issue in the case of maps as there is no negative distance between two nodes [7]. This algorithm is faster as compared to the Bellman-Ford algorithm because it uses a priority queue that greedily selects the closest node that has not been visited and calculates the distance of selected nodes only.

IV. Floyd Warshall's Algorithm

Floyd Warshall algorithm finds the shortest path between every pair of vertices. It is similar to the single-source shortest path Dijkstra's algorithm but Dijkstra's algorithm finds out the path from one of the source vertices [8]. If there are n number of nodes then Dijkstra will take time for a single vertex and Floyd Warshall will run the same problem for times for all the nodes. Hence compared to Dijkstra's algorithm it takes more time to find the shortest route. However, this can be implemented for both positive and negative nodes.

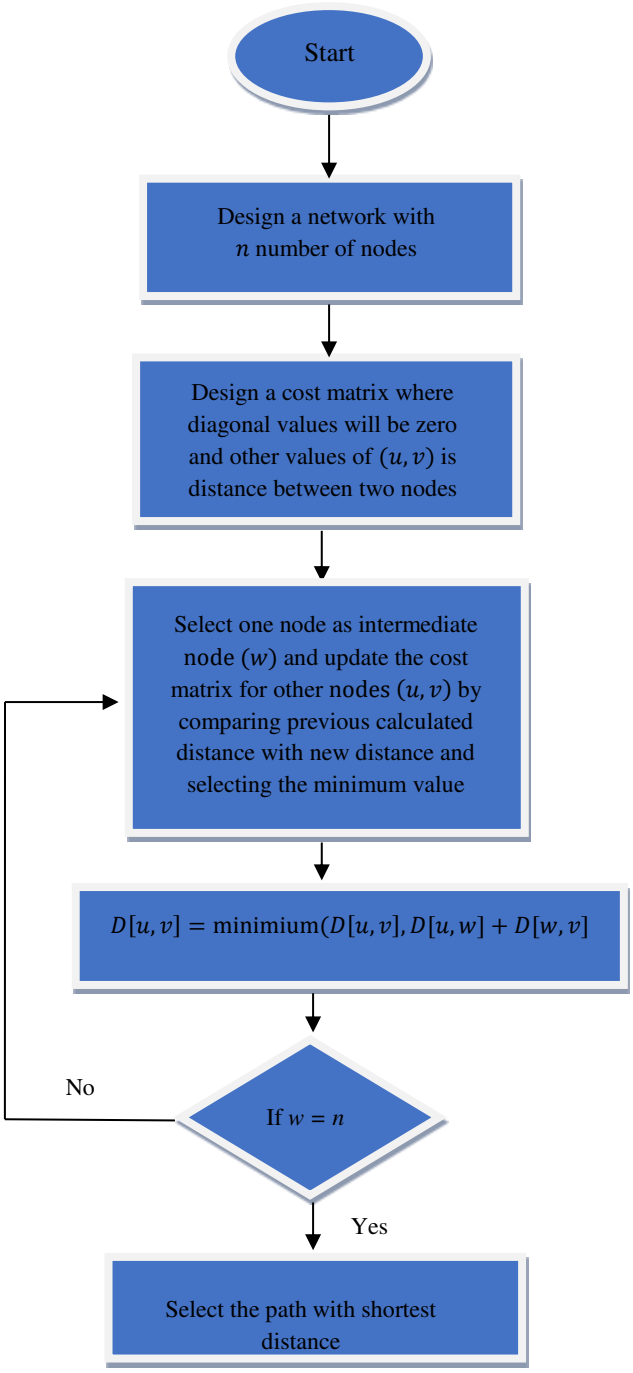

Fig 5. Flow Chart of Floyd Warshall's algorithm

Table 1. Comparison of the techniques discussed to calculate shortest distance in WSN

S.No.	Technique	Features	Advantages	Disadvantages
1.	Divide and Conquer Technique [5]	Recursive Method, parallel processing of data, Time complexity is $\theta(n)$, Recurrence relation for tracing of data is given as: $T(n) = n + 1$	Can be used for layout of networks that are simple in design	Cannot be used for complex networks, Duplicating of sub-networks, Low efficiency
2.	Bellman Ford's Algorithm [6]	Dynamic programming strategy, single source shortest path algorithm, Time complexity for complete network is $\theta(n^3)$	Can be used for complex networks	If algorithm fails to update the weights in case of cyclic network then it becomes slow and time consuming
3.	Dijkstra's Algorithm [7]	Single source shortest path algorithm, Greedy method, Time complexity for a complete network is $\theta(n^2)$	Works for both directed and undirected networks, Faster as compared to the Bellman-Ford algorithm	Less surety to work for networks with negative weights.
4.	Floyd Warshall Agorithm [8]	Dynamic programming strategy, Time complexity for a complete network is $\theta(n^3)$	Works for both directed and undirected networks, Can be implemented for both positive and negative nodes.	Takes more time to find the shortest route as compared to Dijkstra's algorithm

The System Model

In this paper, Dijkstra's algorithm is the adopted technique to calculate the shortest distance between the source and destination node and is simulated with CupCarbon U-One 4.2 [9]. The simulator facilitates, envision, debug, and also verify distributed algorithms for academic and scientific projects. The Network is designed by using Open Street Map (OSM) framework where sensors are deployed directly on the map as shown in Fig 6.

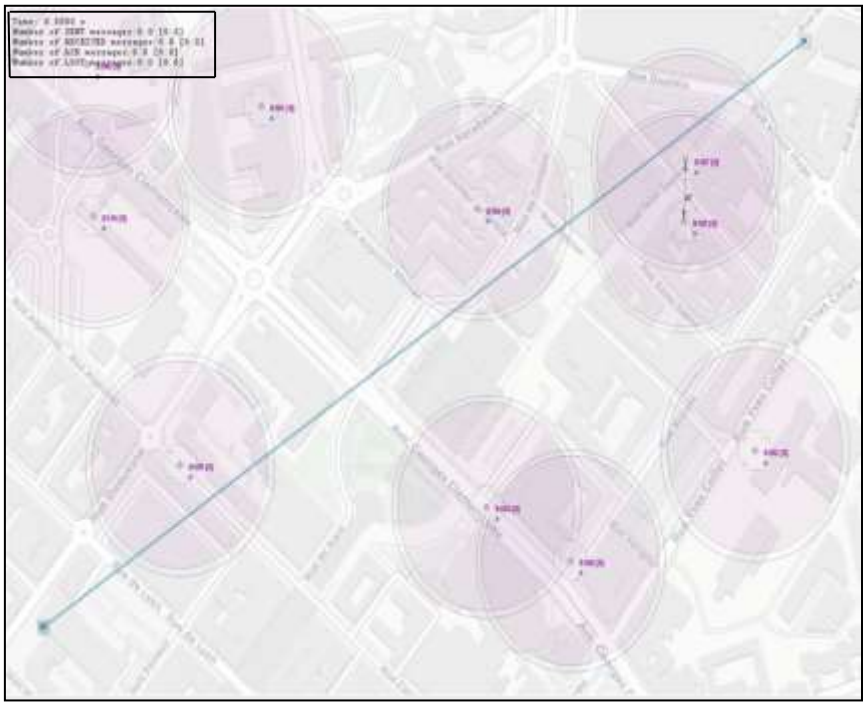

Fig 6. Random deployment of Sensor Nodes

The simulator has its script known as SenScript to program and design each sensor node. In this paper, two cases are considered, where the distance between the source node and destination node is calculated with and without using Dijkstra's algorithm. These two cases are simulated to compute the end-to-end delay, for each case, it is calculated using the mathematical formula as listed in equation (1)

$$\frac{N \times L}{R} \times P \qquad (1)$$

Here P denotes the number of packets sent over a network, N denotes the number of links between the nodes, L denotes the length of the packet and R is the transmission rate. The end-to-end delay for every simulation is computed by the simulator as a default setting and the number of sent messages, number of received messages, number of acknowledged (ACK) messages, and number of lost messages are displayed.

The network designed for 10 sensor nodes deployed randomly is shown in Fig 7.

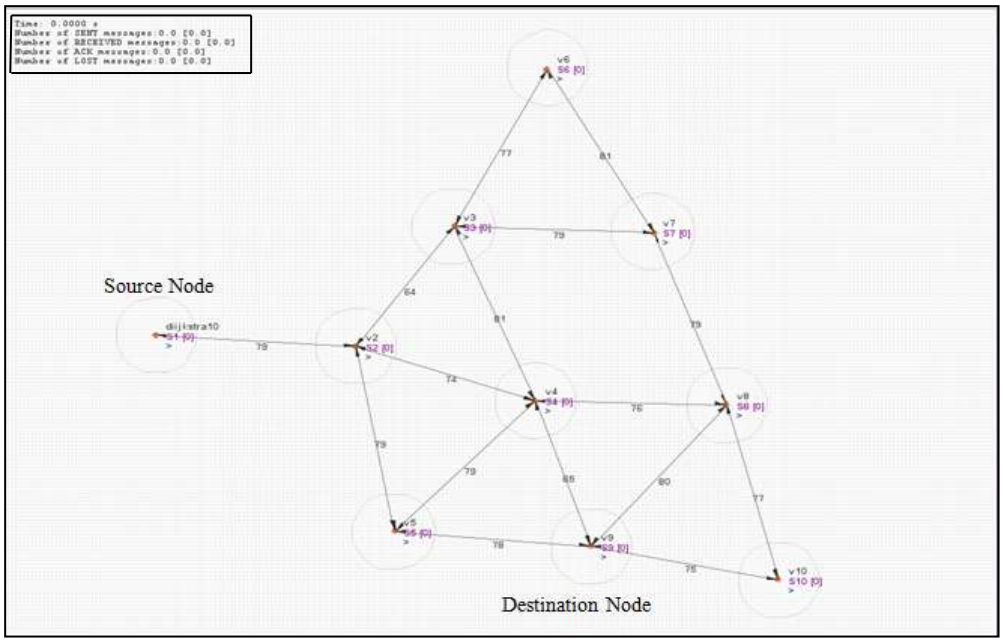

Fig 7. Network of 10 randomly deployed sensor nodes

In this network S_1 is considered as the source node and S_9 as the destination node. Euclidean distance between two nodes can be displayed by the simulator by enabling the 'distances' key in the state bar at the bottom. To reach the S_9 node from S_1 , there exist 14 possible routes that are shown below in Table 2.

Table 2. Path Table for S_1 to S_9 node

<.....Sensor Node Ids.....>

<.....Number of Paths.....>	1	2	3	4	5	6	7	8	9	10	Distances (in meters)
1	1	2	3	7	8	10	9	0	0	0	453
2	1	2	3	4	8	10	9	0	0	0	452
3	1	2	5	4	8	10	9	0	0	0	465
4	1	2	3	7	8	4	9	0	0	0	442
5	1	2	3	7	8	9	0	0	0	0	381
6	1	2	3	4	8	9	0	0	0	0	380

7	1	2	3	4	5	9	0	0	0	0	381
8	1	2	4	8	10	9	0	0	0	0	381
9	1	2	3	4	9	0	0	0	0	0	289
10	1	2	4	8	9	0	0	0	0	0	309
11	1	2	4	5	9	0	0	0	0	0	310
12	1	2	5	4	9	0	0	0	0	0	393
13	1	2	5	9	0	0	0	0	0	0	236
14	1	2	4	9	0	0	0	0	0	0	218

Case1: Calculation of distance between source and destination node without using Dijkstra's algorithm.

Path 3 is selected in this case; The working of the sender and receiver nodes is explained using the algorithm listed below

Step 1: Begin

Step 2: Create an array 'R' of IDs of all the sensor nodes.

Step 3: Run a loop for $i = 1\ to\ N$, where $N$is Number of Nodes in the network.

Step 4: Read the values stored in the array one by one.

Step 5: Check the condition if $R[i]\ != 0$, if true then

Step 6: Send the value of R to neighbouring nodes

Step 7: If above condition is false then repeat step 4 and step 5 till the loop ends

Step 8: End

Step 1: Begin

Step 2: Receive the value of *R* from the neighbouring nodes

Step 3: If the value of *R* is equal to the ID of the current node, then

Step 4: Mark the node indicating that it is the desired node with minimum distance and calculate the distance

Step 5: Otherwise send the route to the neihbours of current node and repeat step 2 and step 3 for *N* number of nodes.

Fig 9. Algorithm for the receiver nodes

By keeping the simulation speed at 500ms and arrow speed at 1000 ms, the algorithm is simulated and the results obtained are shown below

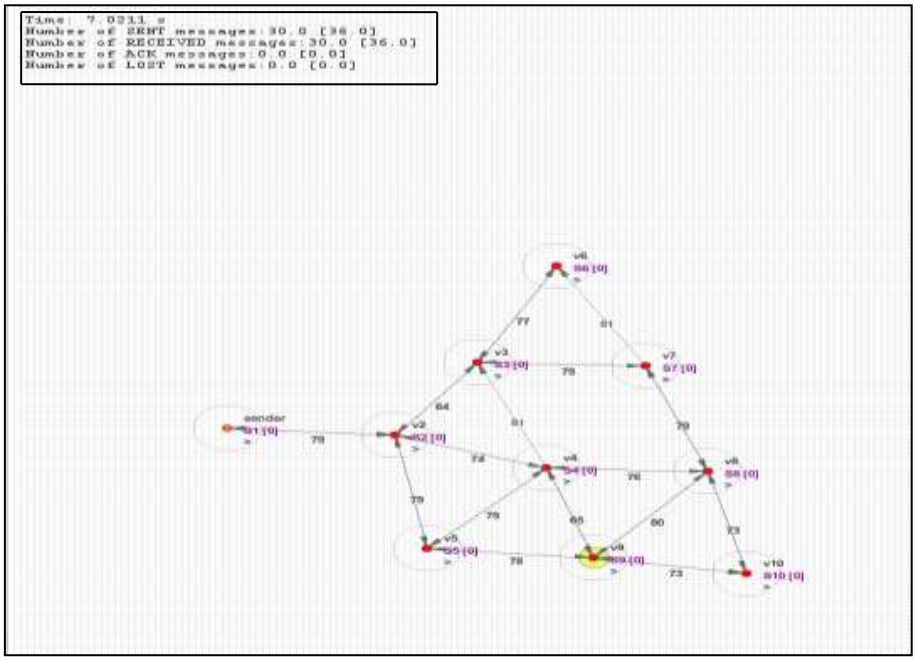

Fig10. Results obtained for Case 1

The flow of data in Case 1 can be represented with spot table shown in Fig 11.

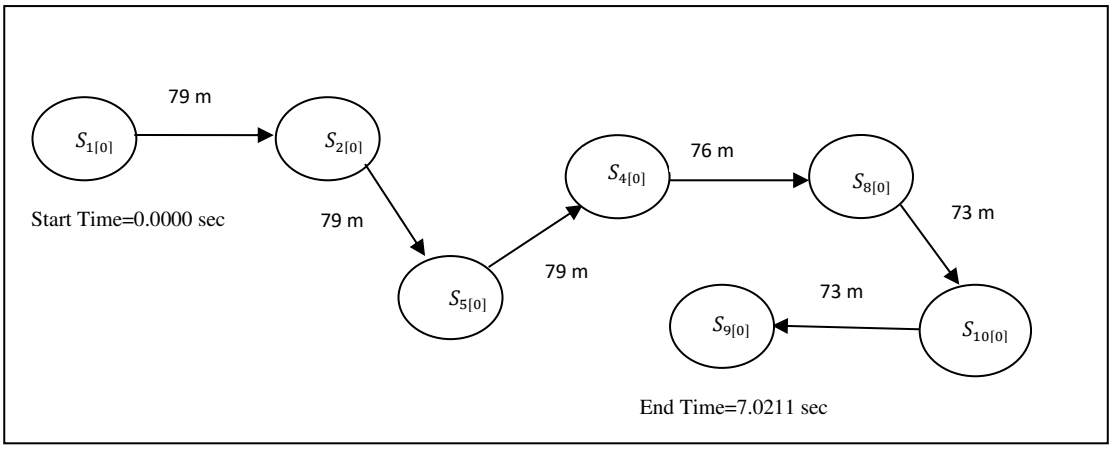

Fig 11. Spot table for Case 1

Energy consumed by the sensor nodes with respect to time in process of sending and receiving data is shown in Fig 12.

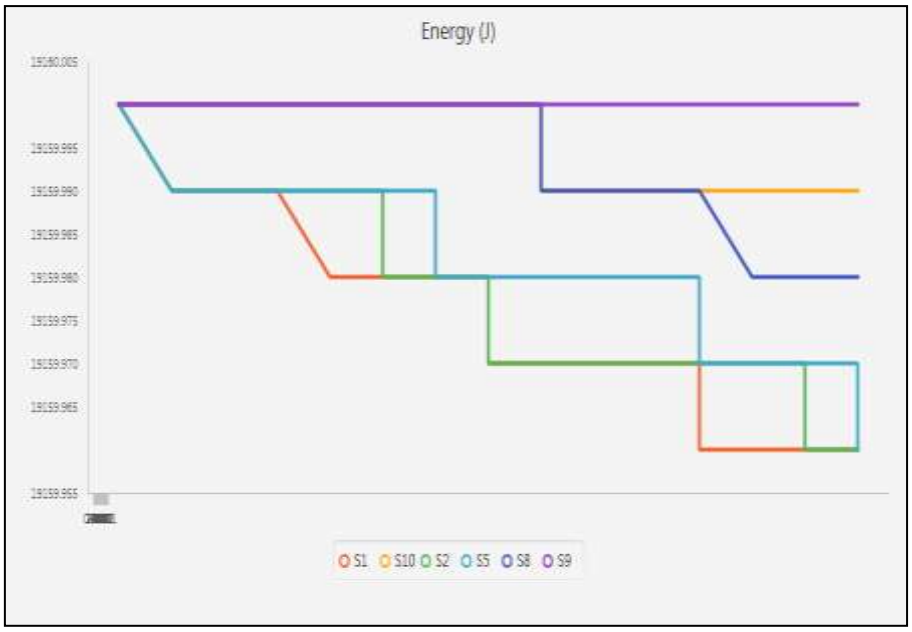

Fig 12. Energy consumed by the sensor nodes in Case 1

Case2: Calculation of distance between source and destination node using Dijkstra's algorithm.

In this case, dijkstra10 code is assigned to the source node which will calculate the shortest distance among all the 14 possible paths and also calculate the end-to-end delay. Working of the sender node and other nodes is explained using the following algorithm in Fig 13.

Step 1: Begin

Step 2: Create a cost-matrix of $N \times N$, where N is the number of sensor nodes

Step 3: Create a Path-matrix of $P \times N$, where P is number of paths possible from Source node to destination node

Step 4: Create an array D to store distance between nodes and set V = ID of destination Node

Step 5: Run a loop for i = 1 to P, set s = 0 and row = 1

Step 6: Run a loop for j = 1 to N

Step 7: Check a condition if value (row) is not equal tovalue (V), then

Step 8: Set column = path [1][j + 1] and set $s = s +$ value of cost matrix [row][column]

Step 9: Store the value of s in D array

Step 10: End the inner loop and set row = column

Step 11: End the outer loop and set minimum = values stored in D array

Step 12: End

Fig 13. Algorithm to calculate distances between source node and destination node

Step 1: Begin

Step 2: Run loop for i = 1 to P

Step 3: Check the condition if D[i] <= $minimum$, then set minimum = D[i] and end the loop

Step 4: Set value of index = i

Step 5: Run a loop for i = 1 to N

Step 6: Check the condition if path-matrix [Index] [i] is not equal to zero, then

Step 7: Send the value of path-matrix [Index] [i] to the neighbouring nodes

Step 8: End

Fig 14. Algorithm to estimate minimum distance

Step 1: Begin

Step 2: Receive the value of route from the neighbouring node

Step 3: If the value of route is equal to the IDs of the current node, then

Step 4: Mark the node indicating that it is the desired node with minimum distance otherwise send the route to the neihbours of current node and repeat Step 2 and Step 3 for N number of nodes

Step 5: End

Fig 15. Algorithm for receiver nodes

By keeping the simulation speed at 500ms and arrow speed at 1000 ms, the algorithm is simulated and the results obtained are shown below in Fig 16.

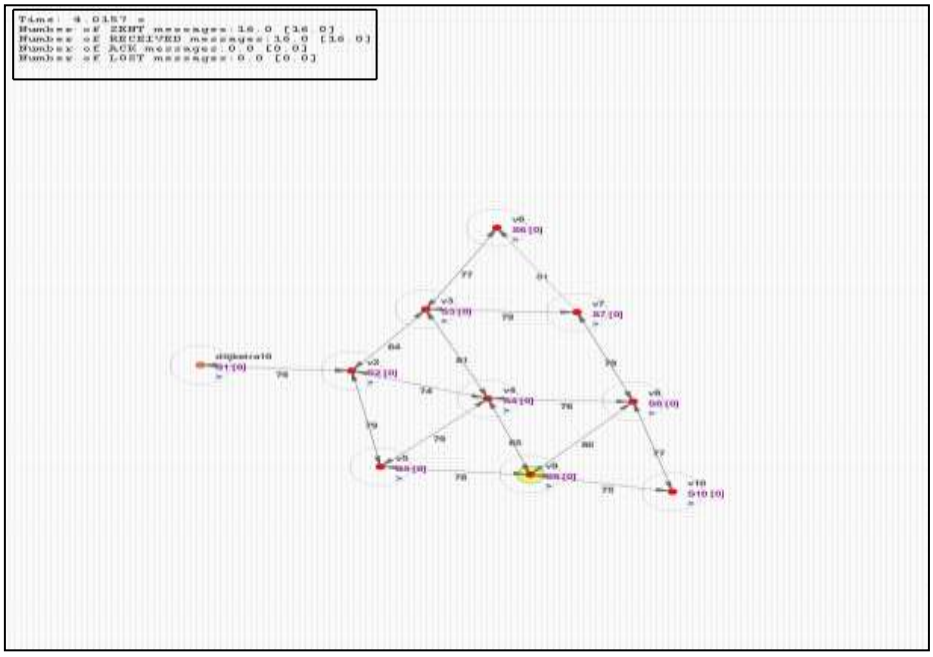

Fig 16. Results obtained for Case 2

The flow of data in Case 2 can be represented with spot table as shown in Fig 17.

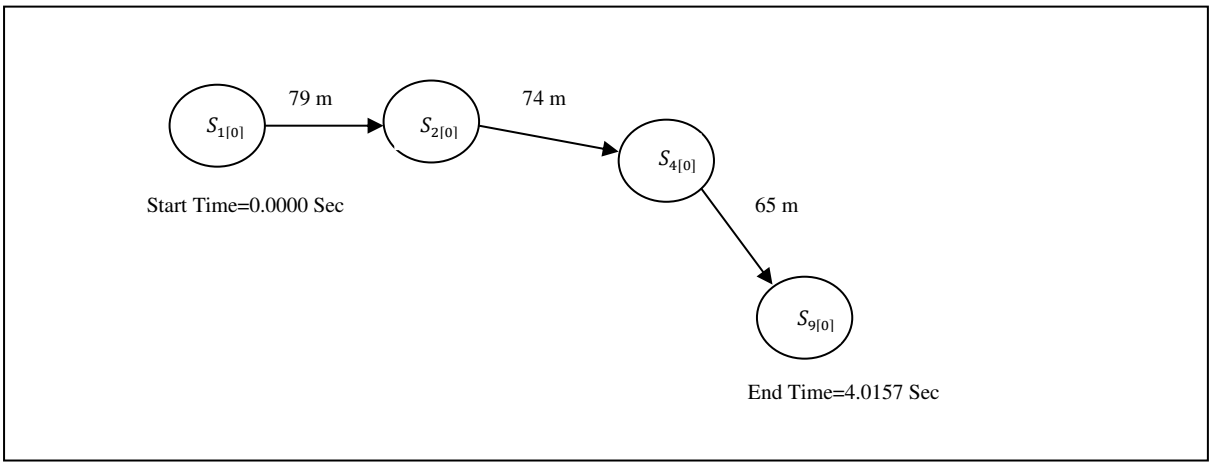

Fig 17. Spot table for Case 2

Energy consumed by the sensor nodes with respect to time in process of sending and receiving data is represented in the graph in Fig.18

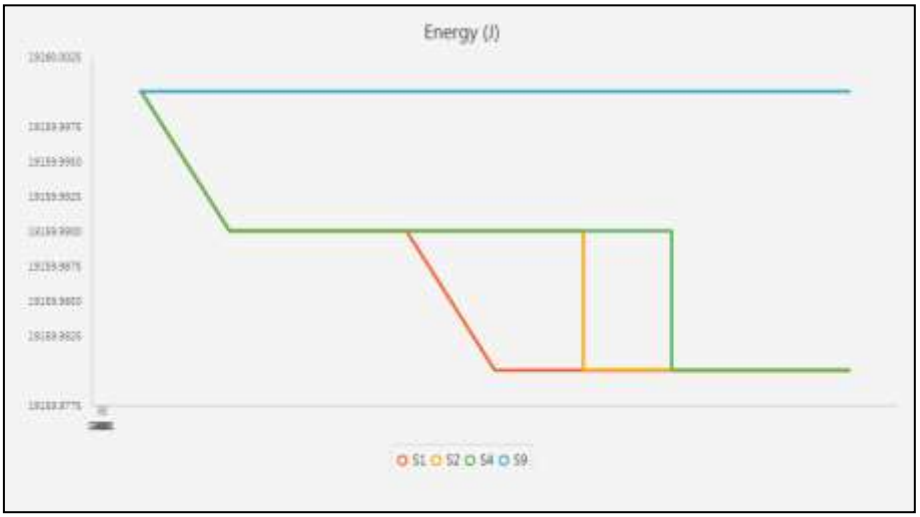

Fig 18. Energy consumed by the sensor nodes in Case 2

The Simulation parameters selected for simulation in both the cases are listed below in Table 3

Table 3. Simulation parameters

Parameters	**Quantity/Value**
Number of Nodes	10
Sensor Radius	20 m

Energy of Sensor Nodes	19159.976772799993 J
Simulation Time	7.0211 s (for Case1) and 4.0157 s (for Case 2)
Simulation Speed	500 ms
Arrow Speed	1000 ms
Number of Iterations	6 (for Case 1) and 3 (for Case 2)
Ambient Condition	Free Space
Agent	Node
Radio Range	100 m
Node Type	Static
Traffic	TCP, FTP
MAC Type	802.15.4
Protocol	Zigbee

RESULT ANALYSIS

The results obtained for the two cases are compared and are highlighted in Table 4

Table 4. Comparison of results

Parameters	**Case 1**	**Case 2**
Sensor nodes used to reach destination node	7	4
End to End delay	7.0211 sec	4.0157 sec
Sent messages	30	16
Received Messages	30	16

Distance between the source and destination node	459 m	218 m

From Table 4 it is observed that the number of sensor nodes used in Case 2 is less as compared to Case 1 which shows that Dijkstra's algorithm is an energy-efficient algorithm that saves the energy of other unused nodes and hence makes the network stabilized and long sustaining. Also, the end-to-end delay for Case 2 is much less as compared to Case 1 which shows that the delivery of the data is faster in Dijkstra's algorithm. The sensor nodes use more energy for communicating with other nodes as compared to the computation of the code and it can be observed from Table 4 that sent and received messages in Case 2 are much less as compared to Case 1 which makes the life span of the nodes longer than in Case 1.

CONCLUSION and FUTURE WORK

In this paper among the many attributes of a smart city i.e., smart mobility is reviewed which can be attained by computing the fastest distance in between the source and destination node. Features of various techniques used to calculate the shortest distance are studied and comparison is drawn to gain knowledge about their advantages and disadvantages for WSN. The technique opted in this paper is Dikstra's algorithm which is simulated with the help of WSN and IoT simulator CupCarbon U-One 4.2. A network of 10 sensor nodes is developed in the simulator and distance in between the source node and destination node is determined considering two cases wherein Case 1 distance is calculated without using Dijkstra's algorithm and in Case 2 distance is calculated using Dijkstra's algorithm. Outcomes obtained in these two cases are compared and it is observed that Dijkstra's algorithm selects the shortest route to the destination node using a smaller number of sensor nodes in the network that ensures the long sustainability of the network. Further, it is observed that in Case 2 using Dijkstra's algorithm end to end delay between the source node and destination node is reduced by 42.80% as compared to Case 1 as a result there is a fast and effective transfer of information in the network and hence has a great impact on network lifetime. Also, simulation results show that in Case 2 using Dijkstra's Algorithm communication carried out in between nodes is reduced by 46.66% as compared to Case 1 using less no of sent and received messages, which results in less bandwidth requirement for communication and it also reduces the redundancy of data.

The Attribute of smart city discussed in this paper i.e., smart mobility may find applications in many fields such as smart exit in which a vehicle caught in some emergency condition can use this technique to find the shortest route to reach a safe place, smart accident management in which an ambulance can use the shortest route possible

to reach the hospital, smart traffic management in which Dijkstra algorithm by incorporating real-time traffic conditions can be used for route planning. For future work, Dijkstra's algorithm can be compared with other techniques discussed in the paper and also can be simulated with CupCarbon to prove its performance. Likewise, its efficiency can be checked on a network by increasing the number of sensor nodes.

Declarations

No funding was received to assist with the preparation of this manuscript.

There is no competing interest.

Data Availability Statement

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

REFERENCES

[1]. Kantarci, B., & Oktug, S.F. (2018). Special Issue: Wireless Sensor and Actuator Networks for Smart Cities. *Journal of Sensor and actuator Networks*, 7, 71-75.

[2]. Martinez, F.J., Toh, C.K., Cano, J.C., Calafate, C.T. & Manzoni, P. (2010). Emergency Services in Future Intelligent Transportation Systems Based on Vehicular Communication Networks. *IEEE Intelligent Transportation Systems Magazine*, 2, 6-20.

[3]. Magray, A., Younis, M. & Sharma, C. (2019). Wireless Sensor Networks Based on Shortest Path Algorithms. *International Journal of Advanced Scientific Research and Management*, 4, 106-110.

[4]. Zhao, F. & Guibas, L. (2004). *Wireless Sensor Networks, an Information Processing Approach*. Morgan Kaufmann Publishers is an imprint of Elsevier.

[5]. Kaleka, G. K. & Singh, H. (2014). Analyze the Performance of Divide and Conquer Scheme for Wireless Sensor Network, *International Journal of Computer Science and Mobile Computing,* 3, 261-265.

[6]. Mummoorthy, A., Bhasker, B. & Kumar, T. J. (2018). Using of Bellman Fords Algorithm in WSN to Identify the Shortest Path and Improve the Battery Power & Control the DDOS Attackers and Monitor the System Environment. *Bonfring International Journal of Networking Technologies and Applications,* 5, 9-11.

[7]. Goyal, M., Kumar, V. & Dahiya, A. (2015). Performance measurement of wireless sensor network using Dijkstra's algorithm. *International Journal of Computer Science & Management Studies*, 18, 2231 – 5268.

[8]. Khan, P., Konar, G. & Chakraborty, N. (2014). Modification of Floyd-Warshall's algorithm for Shortest Path routing in wireless sensor networks. *2014 Annual IEEE India Conference (INDICON)*, 1-6.

[9]. CupCarbon User Guide. Available at: http://www.cupcarbon.com/cupcarbon_ug.html.

[10]. Giner, V.C., Navas, T.I., Flórez, D.S., & Hernández, T.R.V. (2019). End to End Delay and Energy Consumption in a Two-Tier Cluster Hierarchical Wireless Sensor Networks. *Information Journal by MDPI.* 10, 135-164.