ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

OPTIMIZATION OF QUALITY OF SERVICE IN WIRELESS SENSOR NETWORK USING DIJKSTRA'S ALSORITHM AND FLOYD- WARSHALL ALGORITHM

***Lallan Kumar**
* B.Tech(ECE) from (USICT) Guru Gobind Singh Indraprastha University

**** Vikram Singh**
** B.Tech(ECE) from (USICT) Guru Gobind Singh Indraprastha University

ABSTRACT

Wireless Sensor Networks (WSN) are highly distributed self organized systems. WSN have been deployed in various fields. In recent years there has been a growing interest in Wireless Sensor Networks (WSN). Recent advancements in the field of sensing, computing and communications have attracted research efforts and huge investments from various quarters in the field of WSN in this paper we have minimize the constrain related to design issue(quality of services) using Dijkstra's shortest path algorithm and Floyd-Warshall algorithm .

KEY WORDS-*Wireless sensor network, design issue of wireless network, quality of services, dijkstra's algorithm Shortest path , Floyd-Warshall algorithm, weighted graph, application*

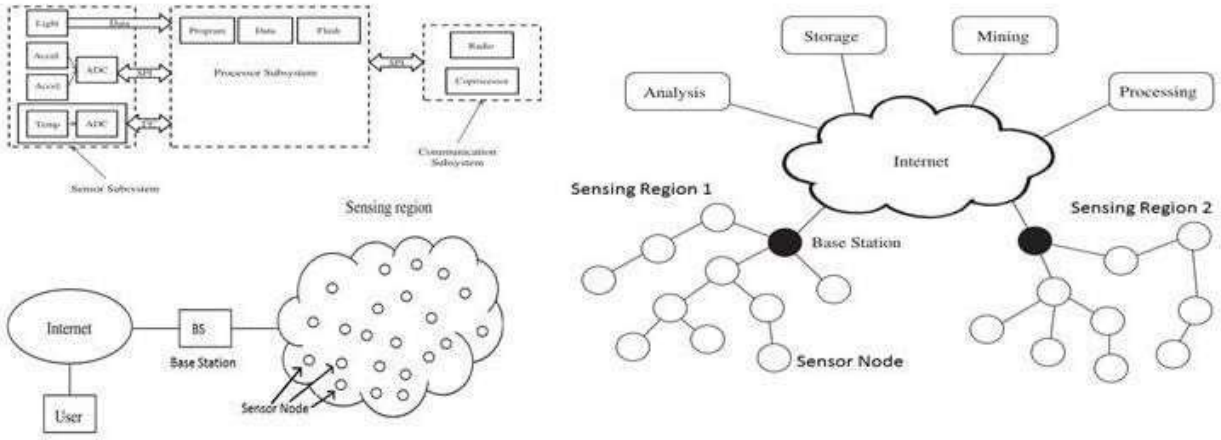
1. INTRODUCTION

A WSN is a collection of wireless nodes with limited energy capabilities that may be mobile or stationary and are located randomly on a dynamically changing environment. The routing strategies selection is an important issue for the efficient delivery of the packets to their destination. Moreover, in such networks, the applied routing strategy should ensure the minimum of the energy consumption and hence maximization of the lifetime of the network . One of the first WSNs was designed and developed in the middle of the 70s by the military and defense industries. WSNs were also used during the Vietnam War in order to support the detection of enemies in remote jungle areas. conditions, where node and link failures are common. The MAC protocol manages radio transmissions and receptions on a shared wireless medium. Therefore MAC has a very high effect on network performance and energy consumption In order to reduce/eliminating the design issue viz.(falt tolerance, scalability, production cost, configuration flexibility, power consumption, data aggregation/fusion, quality of services, and data latency along with overhead are employed using controlled area network and dijkstra's algorithm

By finding shortest path between source node and destination node

In the majority of WSN applications, the connectivity is the main attribute in analysing the quality of service of WSN. It is said that a network is fully connected if each and every node (sensors) can be able to reach the sink node. In geographic routing in WSN is proposed in . A nodecan make routing decisions depending on the geographical location of itself and its nearest nodes. The node first sends a data to the closest node that is nearestto the destination node. This will decrease the value of hop count. When there are no nearest nodes to the destination geographic routing can't optimize the number of hop count. This is called as a Local minimum problem

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188



Picture 1.1: nodes in wireless network

In wireless network data analysis , storage , mining , processing is employed in each and every node before establishing a communication link. In above scenario base station comprises of nodes which sense information as temperature, pressure, humidity ,moisture and send these information to the another base station node via an internet link(wireless networking) On the other hand decryption of data undergoes to avoid unnecessary error within the network.

2. PURPOSED SYSTEM

In this paper we have minimize the constrain related to wireless sensor network (by improving quality of services),here we have establish a minimum cost path from source sensor node to destination sensor node .

We have employed dijkstra's algorithm to calculating shortest path in **single pair wireless network**(viz: single source, single destination) which allow system to choose an optimal path with minimum constrain issue. For **all pair network** have employed Floyd-Warshall algorithm which is used to find **all pair shortest path** problem from a given weighted graph. As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to **all** other nodes in the graph

3. LITERATURE REVIEW

Most of the research on quality of service is in the networking community, especially in distributed multimedia systems. There have been several proposalsand prototype implementations of end-to-end transport protocols for delivering QoS 1guarantees.

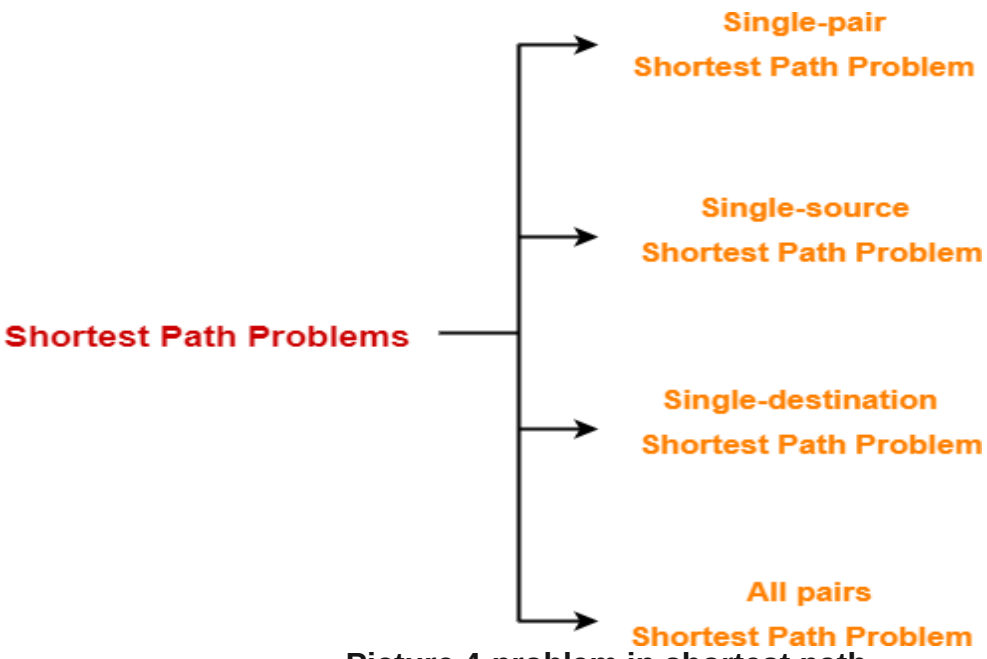
For example, RSVP provides a mechanism for reserving resources along the path froma source host to a destination host so that subsequent data packets are guar- anteed to have certain bandwidth available and meet certain delaybounds.

**Stretching out QoS to remote systems introduces new difficulties because of radio channel qualities, portability the board , higher misfortune, battery influence compels and low transfer speed . Be that as it may, most current QoS conventions can be actualized in remote neighborhood (WLAN) with some change in light of the fact that the last jump is the main remote stage in these systems. In remote systems like Ad hoc remote systems or the new rising remote sensor systems which are absolutely remote, another arrangement of QoS parameters, instruments and conventions are required. In conventional systems, similar to the Internet, the QoS can obtained through the system over-provisioning, traffic building, and differential bundle treatment inside switches, as portrayed in . Customarily, the accentuation is on augmenting start to finish throughput and limiting deferral. Over- provisioning of system assets depends on including gigantic measures of assets in the system. Be that as it may, data transmission accessibility and switch limit are not vast assets and abundance assets are costly, particularly in remote systems

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

4. SHORTEST PATH ALGORITHM IN WIRELESSNETWORK

There are two main types of shortest path algorithms, single-source and all-pairs. Both types have algorithms that perform best in their own way. All-pairs algorithms take longer to run because of the added complexity. All shortest path algorithms return values that can be used to find the shortest path, even if those return values vary in type or form from algorithm to algorithm.



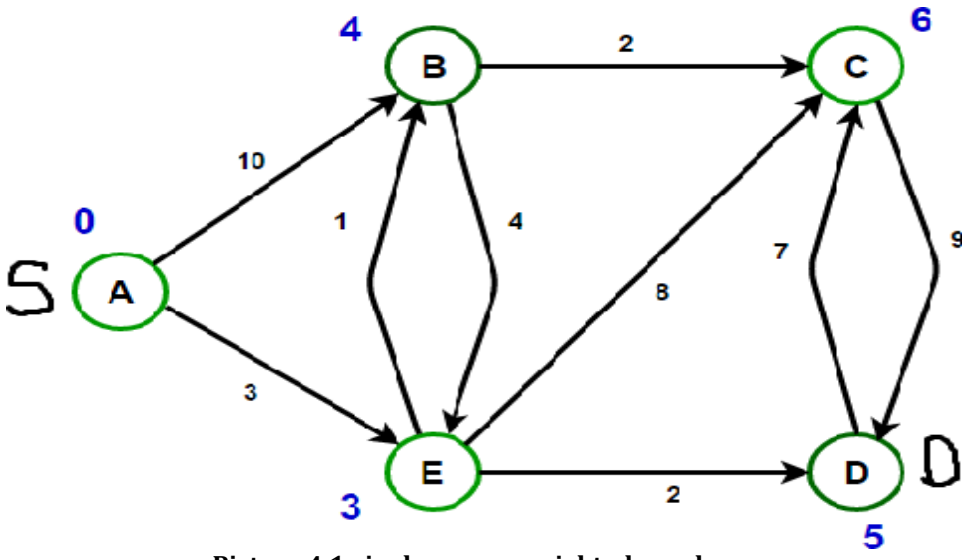
Picture 4:problem in shortest path

4.1) Single-source

Single-source shortest path algorithms operate under the following principle:

> Given a graph GG, with vertices VV, edges EE with weight function w(u, v) =w_{u,v}$w(u,v)=w_{u,v}$, and a single source vertex, ss, return the shortest paths from ss to allother vertices inVV.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188



Picture 4.1:single source weighted graph

The **single-source shortest path** problem, in which we have to find **shortest paths** from a **source** vertex v to all other vertices in the graph. The **single**-destination **shortest path** problem, in which we have to find **shortest paths** from all vertices in the directed graph to a **single** destination vertex v.

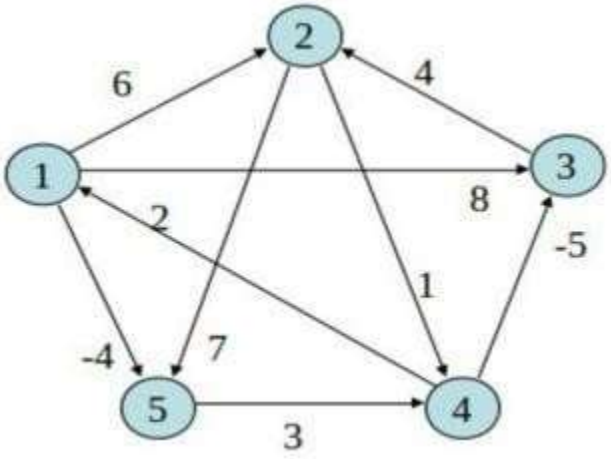
If the goal of the algorithm is to find the shortest path between only two given vertices, s*s* and t*t*, then the algorithm can simply be stopped when that shortest path is found. Because there is no way to decide which vertices to "finish" first, all algorithms that solve for the shortest path between two given vertices have the same worst-case asymptotic complexity as single-source shortest pathalgorithms.

This paradigm also works for the *single-destination shortest path* problem. By reversing all of the edges in a graph, the single-destination problem can be reduced to the single-source problem. So, given a destination vertex, t*t*, this algorithm will find the shortest paths *starting* at all other vertices and ending att*t*.

4.2) All-pairs All-pairs shortest path algorithms follow thisdefinition:

Given a graph G*G*, with vertices V*V*, edges E*E* with weight function w(u, v) = w_{u, v}*w*(*u*,*v*)=*wu*,*v* return the shortest path from u*u* to v*v* for all (u, v)(*u*,*v*) in V*V*.The most common algorithm for the all-pairs problem is the floyd-warshall algorithm. This algorithm returns a matrix of values M*M*, where each cell M_{i, j}*Mi*,*j* is the distance of the shortest path from vertex i*i* to vertex j*j*. Path reconstruction is possible to find the actual path taken to achieve that shortest path, but it is not part of the fundamental algorithm.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188



Picture 4.2:all pair weighted graph

The **all pair shortest path** algorithm is also known as Floyd-Warshall algorithm is used to find **all pair shortest path** problem from a given weighted graph. As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to **all** other nodes in the graph

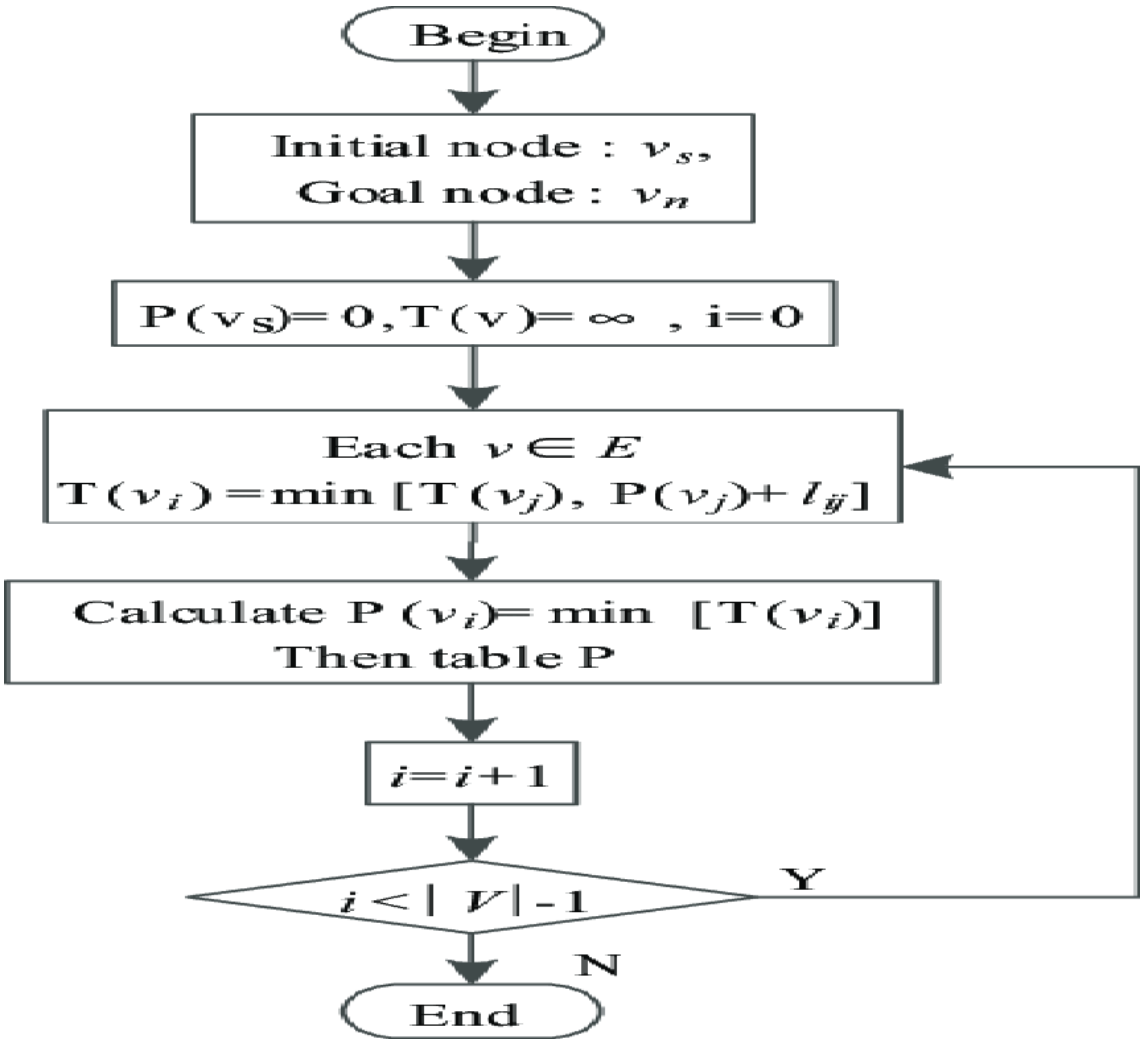
5. DIJKSTRA'S SHORTEST PATHALGORITHM

Dijkstra's algorithm to find the shortest path between *a* and *b*. It picks the unvisited vertex withthe lowest distance, calculates the distance through it to each unvisited neighbor, and updates theneighbor's distance if smaller. Mark visited (set to red) when done with neighbors.

5.1) Steps in dijkstra's algorithm

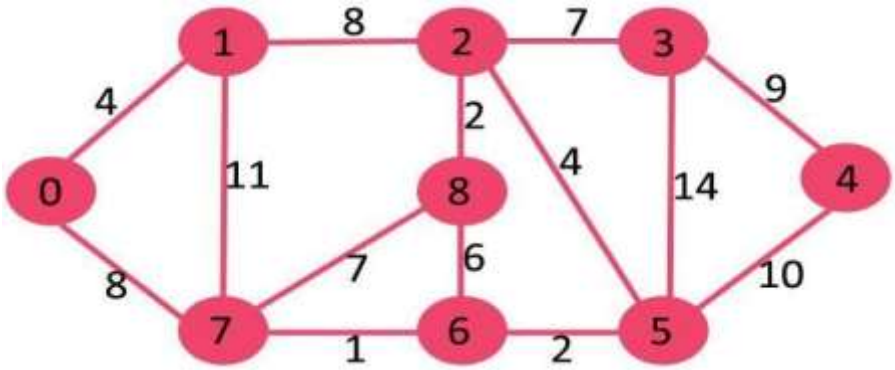
1. Create a set *sptSet* (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized.Initially, this set isempty.
2. Assign a distance value to all vertices in the input graph. Initialize all distancevalues as INFINITE. Assign distance value as 0 for the source vertex so that it is pickedfirst.
3. While *sptSet* doesn't include allvertices
   ….**a)** Pick a vertex u which is not there in *sptSet* and has minimum distance value.
   ….**b)** Include u to *sptSet*.
   ….**c)** Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188



Picture 5.1: flow chart of dijkstra's algorithm for calculating shortest path

Let us understand with the following example:

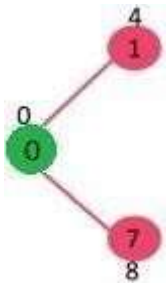


Picture5.1.1:initial graph without hop count

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

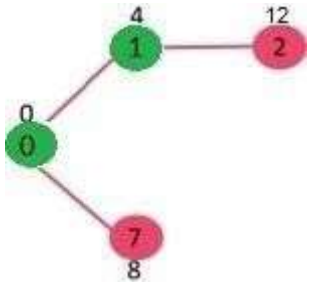
The set *sptSet* is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with minimum distance value.

The vertex 0 is picked, include it in *sptSet*. So *sptSet* becomes {0}. After including 0 to *sptSet*, update distance values of its adjacent vertices. Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8. Following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in SPT are shown in green colour.



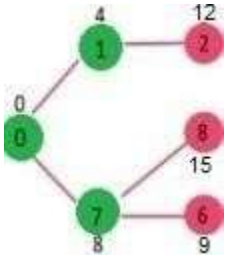
Picture 5.1.2: first hop count

Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex 1 is picked and added to sptSet. So sptSet now becomes {0, 1}. Update thedistance values of adjacent vertices of 1. The distance value of vertex 2 becomes12.



Picture 5.1.3: second hop count

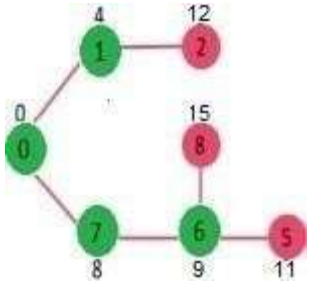
Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 7 is picked. So sptSet now becomes {0, 1, 7}. Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (14 and 9 respectively).



Picture 5.1.4: third hop count

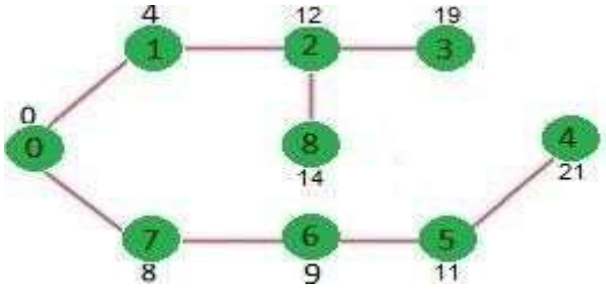
Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 6 is picked. So sptSet now becomes {0, 1, 7, 6}. Update the distance values of adjacent vertices of 6. The distance value of vertex 4 and 8 are updated.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188



Picture 5.1.4: forth hop count

We repeat the above steps until *sptSet* does include all vertices of given graph. Finally, we get the following Shortest Path Tree (SPT).



Picture 5.1.6: shortest route bw source and destination Result

5.2) Code associated dijkstra's algorithms

```
 #include <limits.h>

#include <stdio.h>

// Number of vertices in the graph
#define V 9

// A utility function to find the vertex with minimum distance value, from
// the set of vertices not yet included in shortest path tree
int minDistance(int dist[], bool sptSet[])
{
    // Initialize min value
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

// A utility function to print the constructed distance array
int printSolution(int dist[], int n)
{
```

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

```
    printf("Vertex   Distance from Source\n");
    for (int i = 0; i < V;i++)
        printf("%d tt %d\n", i, dist[i]);
}

// Function that implements Dijkstra's single source shortest path algorithm
// for a graph represented using adjacency matrix representation
void dijkstra(int graph[V][V], int src)
{
    int dist[V]; // Theoutputarray.    dist[i] will hold theshortest
    // distance from src to i

    bool sptSet[V]; // sptSet[i] will be true if vertex i is included in
shortest
    // path tree or shortest distance from src to i is finalized

    // Initialize all distances as INFINITE and stpSet[] as false
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;

    // Distance of source vertex from itself is always 0
    dist[src] = 0;

    // Find shortest path for all vertices
    for (int count = 0; count < V - 1; count++) {
        // Pick the minimum distance vertex from the set of vertices not
        // yet processed. u is always equal to src in the first iteration.
        int u = minDistance(dist, sptSet);

        // Mark the picked vertex as processed
        sptSet[u] = true;

        // Update dist value of the adjacent vertices of the picked vertex.
        for (int v = 0; v < V; v++)

            // Update dist[v] only if is not in sptSet, there is an edge from
            // u to v, and total weight of path fromsrcto    v through uis
            // smaller than current value of dist[v]
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    // print the constructed distance array
    printSolution(dist, V);
}

// driver program to test above function
int main()
{
    /* Let us create the example graph discussed above */
             int graph[V][V] = { { 0 4,  0,   0,  0,  0,  0,  8,  0 },
                                 ,
                               { 4 0,  8,   0,  0,  0,  0,  11,  0 },
                                 ,
```

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

```
                                    { 0 8,  0,   7,  0,  4,  0,  0,  2 },
                                      ,
                                    { 0 0,  7,   0,  9,  14,  0,  0,  0 },
                                      ,
                                    { 0 0,  0,   9,  0,  10,  0,  0,  0 },
                                      ,
                                                  { 0,  0,  4,  14,  10,  0,  2,  0,  0 },
                                                  { 0,  0,  0,  0,  0,  2,  0,  1,  6 },
                                                  { 8,  11,  0,  0,  0,  0,  1,  0,  7 },
                                                   { 0,  0,  2,  0,  0,  0,  6,  7,  0 } };

    dijkstra(graph, 0);

    return 0;
    }
```

Output

```
Vertex   Distance from Source
0 tt 0
1 tt 4
2 tt 12
3 tt 19
4 tt 21
5 tt 11
6 tt 9
7 tt 8
8 tt 14
```

Picture 5.2: output of shortest path

Thus Qos can be computed and enhanced by computing the shortest path by Dijkstra"s algorithm. The node which has the minimum weight is taken to find the finest route from source to destination. The finest route identification resolves the problem of Quality of Service

6. Floyd-Warshallalgorithm

Floyd–Warshall algorithm is an **algorithm** for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles). It does so by comparing all possible paths through the graph between each pair of vertices and that too with $O(V^3)$ comparisons in a graph.

In this algorithm for finding the shortest path between all the pairs of vertices in aweighted graph. This algorithm works for both the directed and undirected weighted graphs. But, it does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative). A weighted graph is a graph in which each edge has a numerical value associated with it. Floyd-Warhshall algorithm is also called as Floyd's algorithm, Roy- Floyd algorithm, Roy-Warshall algorithm or WFIalgorithm.

This algorithm follows the dynamic programming approach to find the shortest path

6.1) Formula for calculation shortest path in anetwork

A^k[I,j]=min{A^k-1[I,j],A^k-1[I,k]+A^k-1[k,j]}

- A^0=threshold matrix which have calculated hop count of all thenodes
- Here A^k[I,j]=required matrix to begenerate

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

- A^k-1[I,j]=previous generated matrix from given thresholdmatrix
- A^k-1[I,k]=new generated matrix which is generated from both threshold matrixand pervious columnmatrix
- A^k-1[k,j]=new generated matrix which is generated from both threshold matrix and previous rowmatrix

6.2) Floyd-Warshall AlgorithmWorking

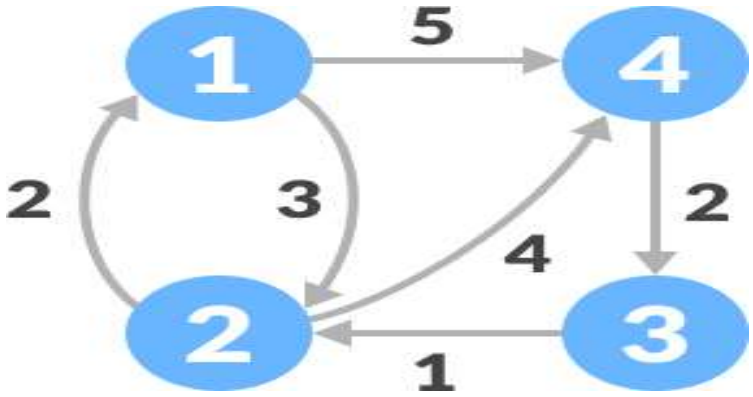
Let the given graph be:

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

Picture 6.2.1: initial graph without path calculation

Step involve in Floyd–Warshall algorithm
Follow the steps below to find the shortest path between all the pairs of vertices.

1)Create a matrix A1 of dimension n*n where n is the number of vertices. The row andthe column are indexed as i and j respectively. i and j are the vertices of thegraph.
Each cell A[i][j] is filled with the distance from the ith vertex to the jth vertex. If there is no path from ith vertex to jth vertex, the cell is left as infinity.

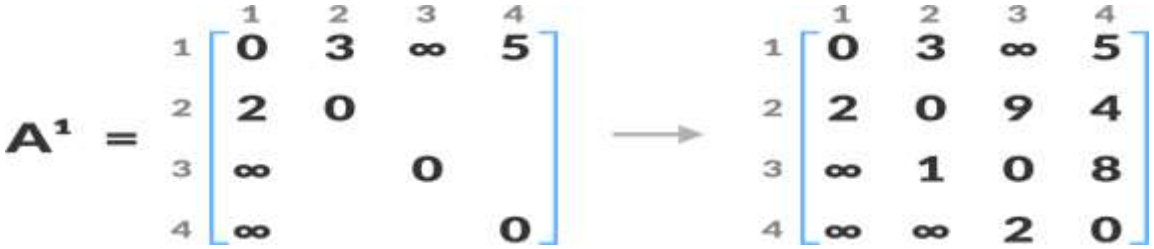


Picture 6.2.2: threshold matrix of network

1) Now, create a matrix A1 using matrix A0. The elements in the first column and the first row are left as they are. The remaining cells are filled in the followingway.

Let k be the intermediate vertex in the shortest path from source to destination. In this step, k is the first vertex.A[i][j] is filled with (A[i][k] + A[k][j]) if (A[i][j] > A[i][k] + A[k][j]). That is, if the direct distance from the source to the destination is greater than the path through the vertex k, then the cell is filled with A[i][k] + A[k][j]. In this step, k is vertex 1. We cacluate the distance from source vertex to destination vertex through this vertex k.

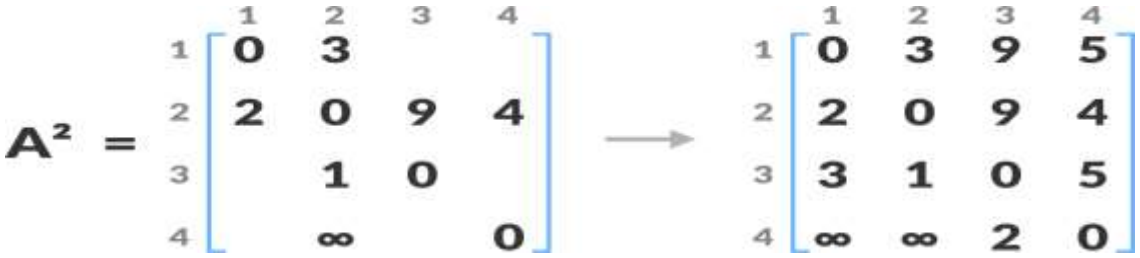
ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188



Picture 6.2.3: first generated matrix A[1,1]

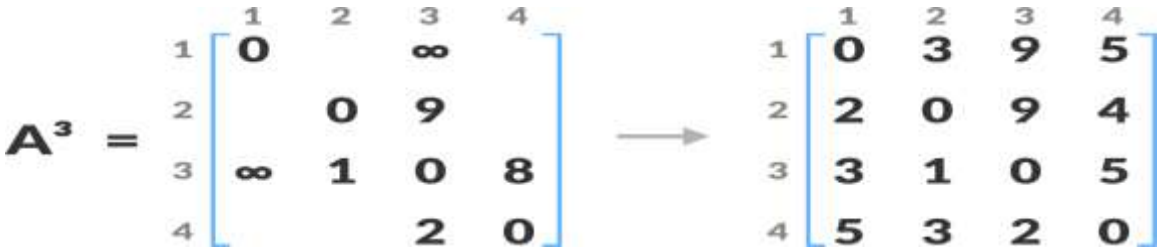
For example: For A1[2, 4], the direct distance from vertex 2 to 4 is 4 and the sum of the distance from vertex 2 to 4 through vertex (ie. from vertex 2 to 1 and from vertex 1 to 4) is 7. Since 4 < 7,
A0[2, 4] is filled with 4.

2) In a similar way, A2 is created using A3. The elements in the second column and the second row are left as theyare.
In this step, k is the second vertex (i.e. vertex 2). The remaining steps are the same as in step 2.

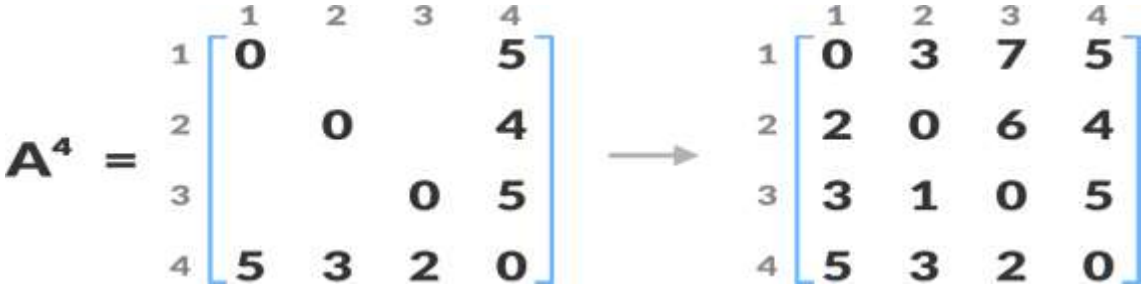


Picture 6.2.4:second generated matrix

3) Similarly, A3 and A4 is alsocreated.



Picture 6.2.4: third generated matrix



Picture 6.2.6: forth generated matrix

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

4) A4 gives the shortest path between each pair ofvertices.

RESULT

Code associated with Floyd–Warshall algorithm

```
// C++ Program for Floyd Warshall Algorithm
#include <bits/stdc++.h>
using namespace std;

// Number of vertices in the graph #define
V 4

/* Define Infinite as a large enough
value.This value will be used for vertices
not connected to each other */ #define
INF 99999

// A function to print the solution matrix void
printSolution(int dist[][V]);

// Solves the all-pairs shortest path
// problem using Floyd Warshall algorithm
void floydWarshall (int graph[][V])
{
   /* dist[][] will be the output matrix that
   will finally have the shortest
   distances between every pair of vertices */ int
   dist[V][V], i, j, k;

   /* Initialize the solution matrix same as
   input graph matrix. Or we can say the
   initial values of shortest distances are
   based on shortest paths considering no
   intermediate vertex.*/
   for (i = 0; i < V; i++) for
      (j = 0; j < V; j++)

         dist[i][j] = graph[i][j];

   /* Add all vertices one by one to
   the set of intermediatevertices.
   ---> Before start of aniteration,
   we have shortest distances between all
   pairs of vertices such that the
   shortest distances consider only the vertices
   in set {0, 1, 2, .. k-1} as intermediate
   vertices.
   ----> After the end of an iteration,
   vertex no. k is added to the set of
   intermediate vertices and the set becomes {0, 1, 2, .. k} */ for
```

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

```
    (k = 0; k < V; k++)
    {
        // Pick all vertices as source one by one for (i
        = 0; i < V; i++)
        {
            // Pick all vertices as destination for the
            // above picked source for
            (j = 0; j < V; j++)
            {
                // If vertex k is on the shortest path from
                // i to j, then update the value of dist[i][j] if
                (dist[i][k] + dist[k][j] < dist[i][j])
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }

    // Print the shortest distance matrix
    printSolution(dist);
}

/* A utility function to print solution */ void
printSolution(int dist[][V])
{
    cout<<"The following matrix shows the shortest distances" "
          between every pair of vertices \n";
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            if (dist[i][j] == INF)
                cout<<"INF"<<"       ";
            else
```

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

```
            cout<<dist[i][j]<<"       ";
        }
        cout<<endl;
    }
}

// Driver code
intmain()
{
    /* Let us create the following weighted graph 10
    (0)------->(3)
        |     /|\
    4|      |
        |     |1
    \|/     |
    (1)------->(2)
            3     */
    int graph[V][V] = { {0, 4, INF, 10},
                        {INF, 0, 3,INF},
                        {INF, INF, 0,1},
                        {INF, INF, INF, 0}
                    };

    // Print the solution
    floydWarshall(graph
    ); return 0;
}
```

output

```
Following matrix shows the shortest distances between every pair of
vertices

      0      4      8      9

    INF      0      3      4

    INF    INF      0      1

    INF    INF    INF      0
```

Thus this matrix show shortest distance between pair of vertices that result in less hop count and eventually reduces the traffic load thus quality of network can be improve.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

7) CONCLUSION AND FUTURE WORK

Thus the quality of service can be upgraded by combining the Dijkstra"s algorithm in single source network and Floyd–Warshall algorithm in all pair source network. Qos can be computed and enhanced by computing the shortest path by Dijkstra"s algorithm. The node which has the minimum weight is taken to find the finest route from source to destination. The finest route identification resolves the problem of Quality of Service. Even though the Qos is solved, it cannot solve the problem of privacy. The privacy, security and integrity constraints can be resolved by SafeQ and extended watchdog algorithm. The security parameter determines the malicious nodes and hence the attacks can be prevented. Thus the projected method will ensure the optimal Quality of Service is achieved.

There are other types of quality of services issue in wireless network that yet to research along with hardware simulation for same.

We also believe that Future investigations will focus on extending our algorithm to the multihop situation. Besides, exploring relationship between the CWmin and the access probability in different traffic patterns as well as other effective ways to estimate the network conditions more accurately is also an important future work as well The privacy, security and integrity constraints which can be resolved by SafeQ and extended watchdog algorithm yet to explore The security parameter determines the malicious nodes and hence the attacks can be prevented

8) APPLICATION OF WIRELESSNETWORK

8.1) MILITARYAPPLICATIONS

Due to the self-organization, rapid deployment and fault tolerance characteristics of wireless sensor networks, they are useful in monitoring friendly forces, arms and ammunition; target tracking; battle damage assessment and nuclear, biological and chemical attack detection and reconnaissance.

8.1.1) Target Tracking: Sensor networks can be incorporated into guidance systems of the intelligent ammunitions for tracking the targets insea.

8.1.2) Battle damage assessment: To gather the battle damage assessment data, sensor networks can be deployed in the battle field before and after theattacks.

8.2) ENVIRONMENTAL APPLICATIONS: The environmental applications of sensor networks include tracking the movements of birds and animals; monitoring environmental conditions that affect crops and livestock; precision agriculture; biological and environmental monitoring in marine, soil , forest fire detection; and flood detection.

8.2.1) Tracking the movements of birds, small animals, andinsects:

To perform a biological study of the habitats of birds and animals, sensor networks can be used to collect reports at regular intervals and further integrated to study their life cycle.

8.2.2)Monitoring environmental conditions that affect crops andlivestock:

To enhance the agricultural productivity, it is necessary to detect the various factors that affect crops and livestock. Sensor nodes monitor the environmental conditions that can influence their growth and accordingly design measures to overcome it.

8.3) HOMEAPPLICATIONS

Smart sensor nodes can be embedded in electrical appliances, such as vacuum cleaners, micro-wave ovens, refrigerators, VCRs and air conditioners. These sensor nodes can interact with each other and can be remotely controlled and monitored

9) REFERENCE

1. *Basaran C, Kang KD. In: Misra S, Woungang I, Misra SC, Eds. Quality of Servicein Wireless Sensor Networks. London: Springer 2009:305-17.*
2. *E. Crawley, R. Nair, B. Rajagopalan and H. Sandick, 1998. A Framework forQoS-Based Routing in the Internet,RFC2386.*
3. *[3] Sohrabi, K., Goa, J., Ailawadhi, V., and Pottie, G. J. 2000. Protocols for self-organization of a wireless sensor network. IEEE Personal Commun 7, 5 (Oct.), 16--27.*
4. *[4] X. Huang and Y. Fang, "Multiconstrained QoS Mutlipath Routing in Wireless Sensor Networks," Wireless Networks, Vol. 14, No. 4, 2008, pp. 465-478.*
5. *[5] D. C. Hoang, P. Yadav, R. Kumar and S. Panda, "Real-time implementation of a harmony search algorithm-based*

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 10 | October 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

clustering protocol for energyefficient wireless sensor networks", IEEE transactions on Industrial Informatics, vol.10, no.1, pp.774–783, 2014.

6. *[6] J. Niu, L. Cheng, Y. Gu, L. Shu and S. K. Das, "R3E: Reliable reactive routing enhancement for wireless sensor networks", IEEE Transactions on Industrial Electronics, vol.10, no.1, pp.784– 794, 2014.*
7. *[7] P. Desnoyers, D. Ganesan, H. Li, and P. Shenoy, "Presto: A predictive storage architecture for sensor networks," in Proc. 10th HotOS, 2005.*
8. *M. Narasimha and G. Tsudik, "Authentication of outsourced databases usingsignature aggregation and chaining," in Proc. DASFAA,2006.*
9. *K´alm´an Graffi, Parag S. Mogre, Matthias Hollick, and Ralf Steinmetz , "Detectionof Colluding Misbehaving Nodes in Mobile Ad hoc and Wireless Mesh Networks," in IEEE GLOBECOM, November2007.*
10. *Extended Watchdog Mechanism for Wireless Sensor Networks Lei Huang, LixiangLiu, Journal of Information and Computing Science,2007.*
11. *6-Youngho Cho and Gang Qu, Insider Threats against Trust Mechanism withWatchdog and Defending Approaches in Wireless Sensor Networks, IEEE Symposium on Security and Privacy Workshops,2012.*
12. *Garcia-Macias, A., Rousseau, F., Berger-Sabbatel, G., Toumi, L., and Duda, A. Qualityof service and mobility for the wireless internet. Wireless Networks 9, 4 (2003),341–352.*
13. *Mahadevan, I., and Sivalingam, K. M. Quality of service architectures forwireless networks: Intserv and diffserv models. In ISPAN (1999), pp.420–425.*
14. *Vali D, Paskalis S, Kaloxylos A, Merakos L. A survey of internet QoS signaling.IEEE Commun Surv Tutorials 2004; 6:32-12.*