International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P), Volume-5, Issue-1, May 2016

Shortest Path Determination in a Wireless Packet Switch Network System in University of Calabar Using a Modified Dijkstra's Algorithm

Ofem O. A., Edim E. A., Ele S. I.,

***Abstract*— The problem of finding the shortest path between two modes is a well known problem in network analysis. Optimal routing has been widely studied for interconnection network. This research considers the problem of finding the shortest path in a wireless packet switch network system in the University of Calabar environment, Calabar, Nigeria. Its theoretical approach, implementation and application. First a historical background of sp algorithm was given and basic concepts of network analysis in connection with congestion and delay issues were explored for the later use. The report was concentrated on the modified Dijkstra's algorithm and the open shortest path first (OSPF) algorithm, use in finding single source shortest path in a one-to-all undirected wireless packet switch network system in the University of Calabar environment, Calabar, Nigeria, with distance costs of links in order of O (n+m) time, where n is the number of nodes and m is the number of links in the network. The algorithm was presented in the context of wireless packet switch network system in the University of Calabar, Calabr, Nigeria. The aim of the research is to designed a wireless packet switch network in the University of Calabar environment with the objective of determining the shortest path, dij taken by a message (packet) to traverse from a source node (current) to a destination (sink) node considering four different routes in the network system.**

***Index Terms*— Graph, transit distance (dij) wireless packet switch network adjacency list, adjacency matrix.**

I. INTRODUCTION

In this research, our network model is abstracted as a graph G (V, E). In graph theory, the shortest path problem is used to determine the path between source vertices and destination vertices (or nodes) such that the sum of the weights of its constituent edges is minimized.

A graph G is a collection of two sets V and E where V is the collection of vertices Vo, Vi, --- Vn-1also called nodes and E is the collection of edges e1, e2, ----, en where an edge is an arc which connects two nodes (Delling, 2008).

This can be represented as:

G = (V, E)

V (G) = (Vo, V1, ----, Vn) or set of vertices

E (G) = (e1, e2, en) or set of edges

In a graph structure, we have two major components namely: nodes and edges. This we need to design the data structure to keep these components in memory. There are two ways of representing the graph in computer memory. First one is the sequential representation and second one is the linked list representation. Adjacency matrix is the matrix, which keeps the information of adjacent nodes. In order words we cannot say that this matrix keeps the information that whether this node is adjacent to any other node or not. If the adjacency matrix of the graph is sparese then it is better to represent the graph through adjacency linked list. Our network model in the University of Calabar is a sparese wireless, undirected packet switch network thus we represent the graph using adjacency priority linked list. In this adjacency link list representation of the graph, we will, maintain two lists. First list will keep information of all nodes in the graph and second list will maintain a list of adjacency nodes for each node. There are several cases in graph where we have a need to know the shortest path from one node to another node. In our wireless packet switch network, the switches or routers constitute the nodes, while the distances between any two switches constitute the edges of the network. Our problem definition is to find the shortest path (dij) that a packet will take to traverse from a source node to a destination node. Our research scenario is a network routing concept.

There can be several paths for the packets to go from one node to another node. But the shortest path is that path in which the sum of weights of the included edges is the minimum. There are several algorithms to determine or find the shortest path in a network graph. Here we describe the modified dikjstra's algorithm (Ahn, Ramakrishna, Rang & Choi, 2001).

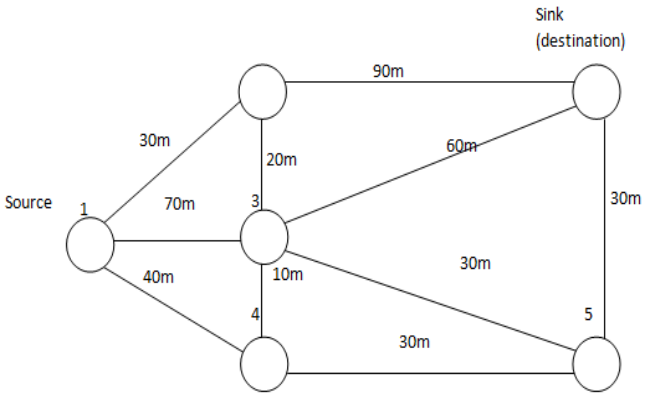


Fig. 1.0 A wireless packet switch network modeled as a graph structure in the University of Calabar.

From the above graph structure, our objective is to find the shortest path that a packet can traverse from the source node to the destination node. The source node is node 1and the destination node is node 6.

We label each node with transit distance or distance cost dij,

Ofem O. A., Department of Computer Science University of Calabar, Nigeria
Edim E. A., Department of Computer Science University of Calabar, Nigeria
Ele S. I., Department of Computer Science University of Calabar, Nigeria

predecessor, and status. Distance of node represents the shortest distance of that node from the source node, and predecessor of node represents the node which precedes the given node in the shortest path from the source. Status of a node can be permanent or temporary, making a node permanent means that it has been included in the shortest path. Temporary nodes can be relabeled if required but once a node is made permanent it cannot be relabeled.

The procedure for our modified diskstra's algorithm

1. Initially make source node permanent by assigning to it a distance value of zero and make it the current working node. All other nodes are made temporary, nodes that are directly reachable from node. 1 have their temporary label equal to dij while nodes that are not reachable from node 1 have to infinity as their temporary labels.
2. Examine all the temporary neighbours of the current working node and after checking the condition for minimum weight, relabel the required nodes
3. From all the temporary nodes, find out the node which has minimum value of distance, make this node permanent and now this is our current working node.
4. When there is a tie in steps 2 and 3 choose any one but exactly once.
5. Repeat steps 2,3 and 4 until destination node is made permanent.

Advantages of our modified diskstra's algorithm

1. With this algorithm, we can minimize our cost while bulding a network. This is because the modified dijkstra's will find the shortest path value from a given source node to the destination node. Therefore, we need not build much of routers to build path from a node to other.
2. With this algorithm, we can also maximize the efficiency of the system, since it will find out the minimum path value. Recall that path weight is propagation delay for a system.

II. Proposed Work

There are several cases in graph where we need to know the shortest path from one node to another node. The following domains of application of our algorithm follow this approach: Robot path planning, logistic distribution, link-state routing protocol, general electric supply system, water distribution, railway track system, driving direction on websites, like map quest or Google map, plant and facility layout arising in VLSI design, transportation, and the traveling salesman problem (Anjali, Datta & Joshi, 2016). In our wireless packet switch network, our algorithm is very useful in the network for routing concepts. There are several paths for going from one node to another node. But the shortest path is the path in which the sum of weights of the included edges is the minimum. There are several algorithms to find out the shortest path. Here we will describe the modified dijkstra's algorithm.

We implement a total different concept to find out shortest path using the scaled map of the University of Calabar. The map which we are using is scaled map which return correct distance between two switches or routers in our network. Here we will use link list to store and traverse N nodes or vertices. We label each node (switch) with distance predecessor and status. Distance of node represents the shortest distance of that node from the source node, and predecessor of node represents the node which precedes the given node in the shortest path from the source node, status of a node can be permanent or temporary.

III. Research Design Problem and Formulation

Earlier researchers have performed various operations on Dijkstra's algorithm to determine the shortest path between the nodes and have got the better results in their researches for the specified number of nodes. But, the results were limited to the number of nodes fixed at the time while declaring the size of the data structure.

Anjali, Datta & Joshi, (2016) referring to Fuhao ZHANG, introduces the classical dijkstra's algorithm in detail, and describes the useful process of implementation of the algorithm. It describes the adjacent node algorithm which is a better optimization algorithm based on Dijkstra's algorithm. This algorithm makes correlation with each node in the different network topology and information, and avoids the use of co-related matrix that contains huge infinite value, and making it more reliable and suitable analysis of the networks for mass data (). It is prove that this algorithm can save a lot of memory space and is more reliable to the network with huge nodes, but it was found that as node grew larger this approach gets slow in searching nodes.

In this research, our design problem is to design a wireless network system in the University of Calabar, Calabar, Nigeria, with the objective of determining the shortest path from the source, (current) node to the destination (sink) node in an undirected network system. Thus our model posses the following attributes:

1. It is a single source shortest path problem in a one-to-all network.
2. The edge lengths are non-negative
3. The graph is an undirected one
4. The shortest path is computed using just comparison and addition model.

The attribute of a node in the network are:

1. Its distance value, and
2. Its status label

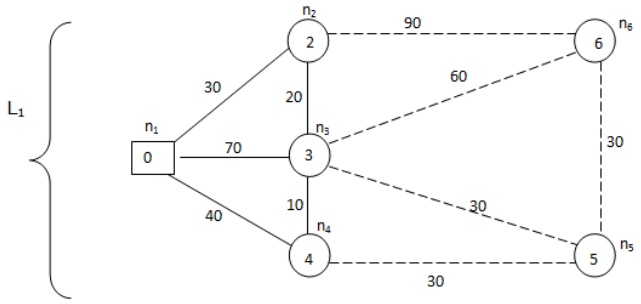
From our network design model in fig. 1.0, the model of research is designed using the linked list priority queue data structure. The simulation is done at different levels of priorities. The priority levels are as follows;

At level 1:

Node 1 ← 0

Reachable nodes from node 1 are nodes 2, 3 and 4. The temporary labels for these nodes are their direct distances from node 1 to the node in question. No temporary label distance is updated here from upper boundary to a lower boundary (Goldberg, Kaplan, & Werneck, 2008). Nodes that cannot be reached directly from node 1 have ∞ as their temporary label. In this model, the researchers used full line for edges that are reachable from the source nodes and broken lines for edges that cannot be reached directly from the source

node. Thus, the node model at this level is:



At level 2:

Compare the temporary labels of n2, n¬3 and n4 and make the smallest of them the current permanent label. No temporary label is updated at this stage.

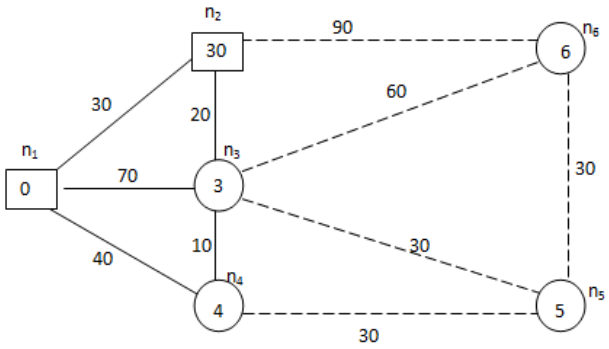
Thus, if n2 < n3 and n2 < n4 then

pL = n2 else pL ≠ n2

∴ Since 30 < 70 and 30 < 40

$n_2 \leftarrow \boxed{30}$

Thus, the logical structure of the model at level 2 becomes:



At level 3:

Update the temporary labels for the nodes that are directly reachable from node 2; compare their TLs and make the smallest TL a permanent label.

The new TL for node 3 = min {∞ , 30 + 20} = 50

min {∞ , 0 + 70} = 70

∴ new TL for node 3 = 50

and new TL for node 6 = min {∞ , 30 + 90} = 120

now compare the TLs and make the smallest of them a permanent label.

If n3 < n4 and n3 < n6 then

pL = n3 otherwise pL ≠ n3

∴ Since the IF statement = False

Another comparison was tested with n4

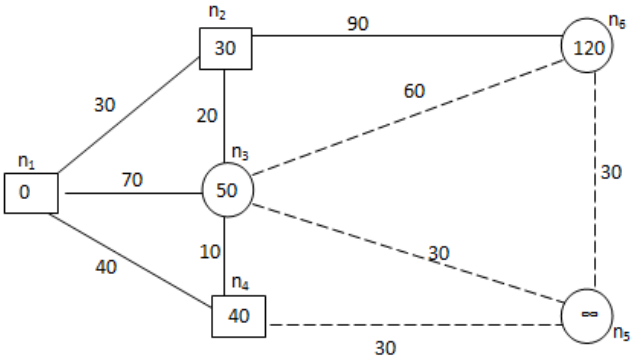
IF n4 < n3 and n4 < n6 then

pL = n4 Else pL ≠ n4

Since 40 < 50 and 40 < 120 then

$n_4 \leftarrow \boxed{40}$

Thus, the logical structure of the model at level 3 becomes:



At level 4:

Update temporary label for node 5:

TL(5) = min {∞ , 40 + 30} = 70

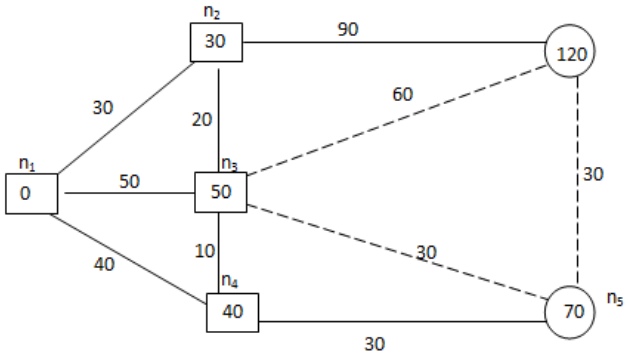
Compare TLS for n3, n5 and n6 and make the smallest a permanent label.

If n3 < n5 and n3 < n¬6 then pL = n3

Else pL ≠ n3 since 50 < 70 and 50 < 120

$n_3 \leftarrow \boxed{50}$

Thus, the logical structure of the model at level 3 becomes:



At level 5:

Update temporary label for node 5 and 6

n5: min{∞ , 50 + 30} = 80

min {∞ , 40 + 30} = 70

TL for n5 = 70

n6: min {∞ , 30 + 90} = 120

min {∞ , 50 + 60} = 110

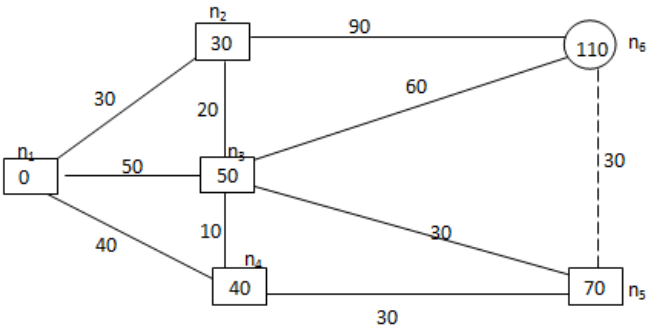
TL for n6 = 110

Now compare TLS for node 5 and 6 and make the smallest a pL.

IF n5 < n¬6 then

$n_5 \leftarrow \boxed{70}$

Thus, the logical structure of the model at level 5 becomes:

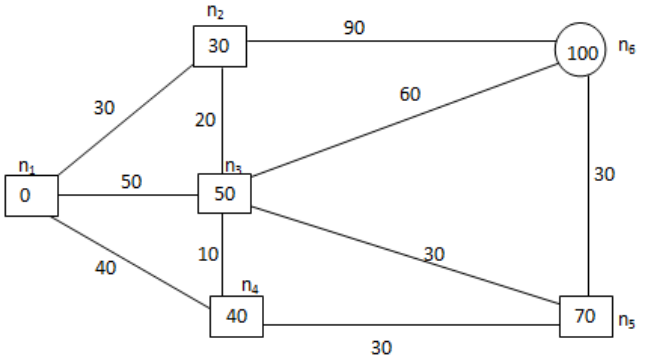


At level 6:

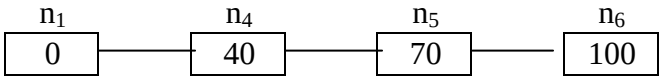
Node 6 is the last node to be visited. Update the temporary label of node 6 and make it permanent.

n6: min {∞ , 50 + 60} = 110

min {∞ , 70 + 30} = 100



Thus, node 6 has a new TL of 100 and it is automatically made permanent. The algorithm converges at this point. The shortest path dij of the network mode is 100 and the path length or link whose dij = 100 is the shortest path of the network. From the logical simulation of the wireless packet switch network model in the University of Calabar, the shortest path for the packets to traverse from the source node to the destination is as follows:



$n_1 \rightarrow n_4 \rightarrow n_5 \rightarrow n_6$

IF TL1 < TL2 < … < T < K

Then pL = TL1 otherwise pL ≠ TL1 subject to further comparison and selection.

IV. Research Formulation

To resolve this problem and to give the best solution for increasing number of nodes we used the priority queue link list having the capacity to store N nodes with some constraints containing the value predecessor, status and distance (Kiran & Ranjit, 2010).

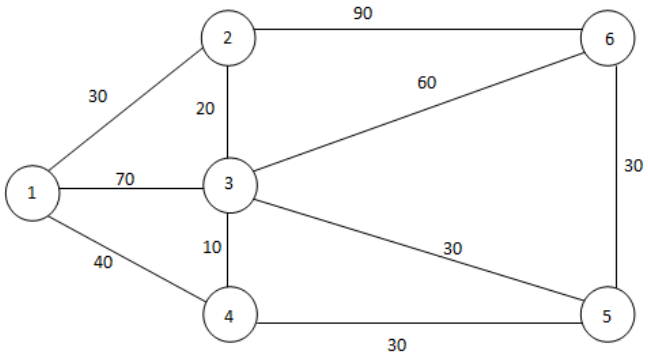
Problem: Given a graph G = {V, E} where V is the set of nodes (or vertices) in G and E the set of edges in G, if n = |V| is the number of vertices in G, then the size of the adjacency matrix required to store G in a computer memory is n x n (or n2):

$$m = \begin{pmatrix} m_{11} & m_{12} & m_{13} \ldots m_{1n} \\ m_{21} & m_{22} & m_{23} \ldots m_{2n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ m_{n1} & m_{n2} & m_{n3} & m_{nn} \end{pmatrix}$$

It is required to represent G in a computer memory with an adjacency matrix of size N<n2.

Solution

Consider a graph G having n = 6 nodes or vertices as shown.



G requires a 6 x 6 (or 36) adjacency matrix to represent it in a computer memory. This representation is shown below.

$$m = \begin{pmatrix} 0 & 30 & 70 & 40 & 0 & 0 \\ 30 & 0 & 20 & 0 & 0 & 90 \\ 70 & 20 & 0 & 10 & 30 & 60 \\ 40 & 0 & 10 & 0 & 30 & 0 \\ 0 & 0 & 30 & 30 & 0 & 30 \\ 0 & 90 & 60 & 0 & 30 & 0 \end{pmatrix}$$

Notice that this adjacency matrix is a symmetric matrix that is MT = M thus, mij = mij ij {1,2,3,4,5,6} and wij = 0 i =j. Hence, if all duplicate is remove, starting from i = j downward, as follows;

30 70 40 0 0
20 0 0 90
10 30 60
30 0
0

Thus, the size of this new matrix is the sum of each row given as 5 + 4 + 3 + 2 + 1 = 15. In general,

$$n - 1 + n - 2 + n - 3 + \ldots n - (n - 1) = \frac{n (n - 1)}{2}$$

Hence, the size of matrix N is

$$N = \frac{n (n - 1)}{2}$$

Where n = number of nodes

Clearly N < n2. in fact.

Theorem: N is less than 50% of n2 that is, N < 50% n2.
Proof:
Suppose N = X% n2, then
$N(n-1) = xn2$
$2 \quad 100, \quad n2 - n = xn2$
$2 \quad 100$

$\Rightarrow 100(n2 - n) = 2xn2 \quad \Rightarrow x = 50(n2 - n) = 50\,n(n-1)$
$n2 \quad n2$

$\Rightarrow 50 - 50 =$
$x \ldots\ldots (1)$
n

Since x depends on n, we can take limit as n · ∞ :

lim x (n) = 50 and hence, the proof:

n · ∞

e.g in the above matrix, n = 6, thus, x (6) = 50 – 50/6 = 50 – 8.33

x(6) = 41.67% ==> 15 is 41.67% of 36

Therefore, percentage reduction = 100 – 41.67 = 58.33%

Instead of creating a multidimensional array M for storing the graph G, we can rearrange the remaining elements of the matrix M as follow;

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
30	70	40	0	0	20	0	0	90	10	30	60	30	0	30

Observe that each element have been isolated with a faint line while each row separated by thick line as shown above. But matrix M is a 2-D array mij, hence, we need to provide an interface for this new one dimensional array so as to make it appear multi-dimensional.

Row Rm and Column Cij.

2-Dimensional array is divided into rows and columns. In the case of matrix mij, i = row and j = column numbers. Hence, we need to find a formula Rm for the row m and Cij for the column C.

$r_1 = 1$	1st row begins at element 3
$r_2 = n - 1 + 1$	2nd row begins at element 2 => 6-1+1
$r_3 = n - 1 + n - 2 + 1$	3rd row beings at element 1 => 6-1+6-2+1
$r_4 = n - 1 + n - 2 + n - 3 + 1$	4th row begins at element 3 => 6-1+6-2+6-3+1
⋮	⋮ " " " " " "
r_m = n-1+n-2+n-3+…+n-(m-1)+1	

$$r_m = (m-1)\,n - \left[\frac{(m-1)\,m-2}{2}\right] \Rightarrow \underline{2n(m-1) - m(m-1) + 2}$$
2

Thus,

$r_m = \underline{(m-1)(2n-m)} + 1 \ldots\ldots (2)$
2

Equation (2) is the required formular for the row number *m*
If *m* = *I*, we have $r_i = \underline{(i-1)(2n-i)} + 1$
2

V. COLUMN

Given a pair of vertices i, j, where i<j, then the column where the weight of the graph G is stored is given by.

$Cij = j - i - 1 \ldots\ldots (3)$
$i<j$

Therefore, given two vertices (or nodes) i and j, the edge from i to j or j to i is given by

$Aij = ri + Cij \ldots\ldots (4)$

Equation (4) is the formular that turns A (a single dimension array) into Aij (a multidimensional array).

Proof:

We must show that Mij = Aji ij {1,2,3,4,5,6}

M12 = 3,

A12 = r1 + C12 ((1 – 1)(2(6) – 1))/2 + 1 + (2 – 1 – 1)

A12 = 0 + 1 +2 – 2 = 1 check the index 1 in A

M35 = 3,

A35 = r3 + C35 ((3 – 1)(2(6) – 1))/2 + 1 + (5 – 3 – 1)

A35 = 12 + 1 = 13 ... check the index 3 in A □

VI. CONCLUSION

Our modified Dijkstra's algorithm is an improved version of the traditional Dijkstra's algorithm. Currently, our algorithm used priority queue with link list with some constraints, plus one modified Dijkstra call. This modified Dijkstra's algorithm should improve the running time of our algorithm by about 58.33% and the shortest path link is node 1 node 4 nodes 5 node 6 where node 1 is the source (current) node while node 6 is the destination (sink) node in our wireless packet switch network system in the University of Calabar, Calabar, Nigeria.

REFERENCES

[1] Anjali, J., Datta, U. and Joshi, K. K. (2016). Modification in Dijkstra's Algorithm to Find the Shortest Path for 'N' Nodes with Constraint. International Journal of Scientific Engineering and Applied Science, 2(1).

[2] Ahn, C. W., Ramakrishna, R. S., Rang, C. G. & Choi, I. C. (2001). "Shortest path routing algorithm using hopfield neural network", Electron. Lett., 37(19): 1176- 1178

[3] Delling, D. & Nannicini, G. (2008). Bidirectional Core-Based Routing in Dynamic Time-Dependent Road Networks. In S.-H. Hong, H. Nagamochi, and T. Fukunaga, editors, Proceedings of the 19th International Symposiumon Algorithms and Computation (ISAAC 08), volume 5369 of Lecture Notes in Computer Science, pp. 813–824.

[4] Delling, D. (2008). Time-Dependent SHARC-Routing. In Proceedings of the 16th Annual European Symposiumon Algorithms (ESA'08), volume 5193 of Lecture Notes in Computer Science, pp. 332–343.

[5] Goldberg, A. V. Kaplan, H. & Werneck, R. F. (2008). Reach for A*: Shortest Path Algorithms with Preprocessing. In C. Demetrescu, A. V. Goldberg, and D. S. Johnson, editors, Shortest Paths: Ninth DIMACS Implementation Challenge, DIMACS Book. American Mathematical Society.

[6] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Algorithm

[7] Kiran, Y. & Ranjit B., (2010). An Approach to Find Kth Shortest Path using Fuzzy Logic. International Journal of Computational Cognition, 8(1)