Modeling and Simulation of Routing Packet Using Dijkstra Algorithm to Achieve the Shortest Routing Path

Ilo S.F, Igbajar Abraham, Paul Kingsley Okah
Computer Engineering Department Michael Okpara University of Agriculture, Umudike
Electronic and Computer Engineering Nnamdi Azikiwe University, Awka
somtoofrancis@gmail.com, igbajar35@gmail.com

Abstract: **For a large number of interconnected autonomous systems consisting of a distinct domain to communicate to different nodes to forward information, routing is a means of doing that, but in order to determine the shortest routing part we need to examine and simulate with Djikstra algorithm. Though there are other algorithm but the advantages of Dijkstra algorithm is that router computes routes independently using the same original status data; they do not depend on the computation of intermediate machines. Because link status messages propagated unchanged, it is easy to debug problems. Because routers perform the route computation locally, it is guaranteed to converge. Finally, because link status messages only carry information about the direct connections from a single router, the size does not depend on the number of networks in the networks in the internet. Thus, Dijkstra algorithms scale better than distance vector algorithms.**

Keywords: **Simulation, Algorithm, Routing, Node, Dijkstra, Network, Quality of service.**

INTRODUCTION

The Routing Algorithm is a process of transmitting data packets from one point to another in a physical network. Various routing algorithm schemes differ in styles and details but the basic functions are similar. To make a connection between source node and destination node, an application usually prepares a flow specification, in which it describes the characteristics of the traffic, and specifies the QoS (Quality of Service) requirements for the flow [Partrdge, 1992]. The task for routing protocol is to select a path to the destination (either a host or a router) that is most likely to satisfy the resource requirements. After that, hop-by-hop negotiation and resource setup can be carried out.

The basic problem for this routing algorithm therefore can be summarized as finding a path in a network for a given set of constraints. Any routing algorithm for solving this problem must however meet the following essential requirements:

- The algorithm must be efficient and be able to scale to large networks such as the internet. Ideally, the complexity of the algorithm should be comparable to current routing algorithms.
- The algorithm must be able to provide sufficient information to make quantitative assessment on resource availability.
- The algorithm must be suitable for current routing architectures, such as distributed hop-by-hop routing.
- The algorithm must be suitable for handling congestion in the network. Taking into consideration the open loop and closed loop solutions [Guerin and Gun, 1992].

1.5 SIGNIFICANCE OF THE STUDY

The world is in the era of communication. This will help the communication industry to improve their services.

- This will aid the manufacturing firms to come up with better designs of routers in future.
- It will encourage research and development in this area in our 3rd world.

LITERATURE REVIEW

There are different routing algorithms available such as BELLMAN FORD ALGORITHM which is decentralized routing algorithm and DIJKSTRA ALGORITHM which is a global routing algorithm. In global routing algorithm, each router has a complete view of the network, whereas in a decentralized routing algorithm each router has a local view consisting of its directly attached neighbors. Out of this Dijkstra algorithm is mostly preferred as it is faster as compared to any other algorithm and its implementation is easier. In this algorithm, router based information that has been collected from other routers. But for Dijkstra algorithm, router computes routes independently using the same original status data; they do not depend on the computation of intermediate machines. Thus, Dijkstra algorithms scale better than distance vector algorithms.

RESEARCH AND IMPLEMENTATION METHODOLOGY

The Dijkstra routing algorithm that was designed follows these sequences of a good software development cycle. Here, search light is on the router and how it works. The router is just a piece of equipment that contains the software used to route packets or messages to their various destinations. This work will achieve an approach used in the design called Dijkstra algorithm and attempt to do some design changes.In order to help achieve this, the java language development platform was chosen. This is as a result of the many resources encountered in exploring the platform. These are some of the properties of the resources: it is simple, object oriented, distributed, robust, secure, architecture neutral, portable, interpreted, high performance, multithreaded, dynamic.

3.3 Flowcharts

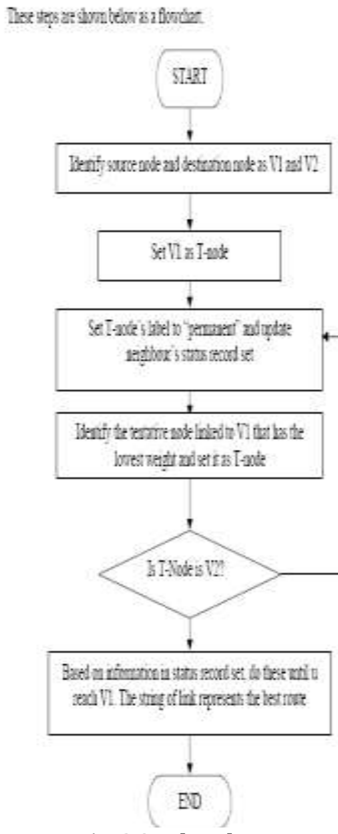

These steps are shown below as a flowchart.
START
Identify source node and destination node as V1 and V2
Set V1 as T-node
Set T-node's label to "permanent" and update neighbour's status record set
Identify the tentative node linked to V1 that has the lowest weight and set it as T-node
Is T-Node is V2?
Based on information in status record set, do these until u reach V1. The string of link represents the best route
END

Fig. 3.2: Flowchart Layout

3.4 DESIGN OF THE SYSTEM

1. Read the number of nodes present in the network.
2. Read the activities between all the nodes.
3. Identity source and destination.
4. Find the distance of all the nodes from the source.
5. Select the node whose distance from the source is the least.
6. Lock the node that has passed i.e. source node in this case.
7. Find the distance of remaining nodes from the selected node and repeat steps 5 and 6.
8. List all the selected nodes from source to destination in the sequence in which they are generated above to give the optimum path.

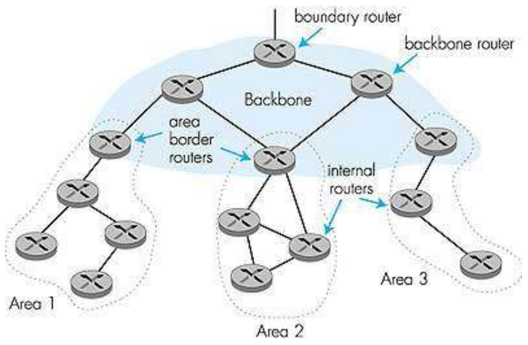


Fig. 3.6: Diagram of a network

3.5 SYSTEM DESCRIPTION

The application is made up of the following classes: The Node class, The Dijkstra class and the Edge class.

The Node class is actually a structure that contains the variables used to store information on the properties of the nodes of the network. Some important variables include the horizontal and vertical position of the node , the link to the previous and succeeding node, the name of the node to mention but a few.

The Edge class is also a structure containing the information used to define the edges connecting different notes. The software uses this information to visually draw the edges on the screen, the initial vertex number, the terminal vertex number, the name of the edge and the distance from the starting edge are all recorded in this class. No methods are necessary since all the processing is done in the Dijkstra class.

3.6 Algorithm for the Dijkstra

The algorithm performs several rules:

Rule1: A graph of the network is built and the adjacency matrix a [i, j] with the weight of links is defined. For the case when a direct link between node Vi and Vj is missing, the weight of the link is assumed as infinity. The source and the destination nodes are noted as NS and NT.

Rule2: A status record set is established for every node with three fields: The first field that shows the previous node, named "predecessor" field. The second filed is named "Length" field and it shows the sum of weights from source to that node. The last field, named "Label" filed, shows the status of the node. Each node can have one status mode: "Permanent" or "Tentative".

Rule3: Initialization of the status record set for all nodes and setting all “Length” to Infinity, and all "Label" as tentative.

Rule 4: Labeling node NS as t node and marking its "Label" as "Permanent". When a label changes to permanent, it never changes again. T node rules as a current chosen node.

Rule5: For all tentative nodes, directly linked to t node, status record set is updated.

Rule6: From all the tentative nodes, choose the one whose weight to NS is less and set it as t node.

Rule7: If this node is not the destination NT, then, go to step 5.

Rule8: If this node is NT, then extract its previous node from status record set and do this until return to NS. The nodes show the best route from NS to NV.

DEVELOPMENT OF THE ALGORITHM.

Development of the code used in implementing ideas start with pseudo code.

Implementing the Dijkstra routing I started with designing the pseudo code.

Pseudo Code:

```
// Pseudo Code for Implementation of Dijkstra Routing Algorithm//
    class node {  /*edge starts from this node*/
    } class edge {          /*defines the edge, initial vertex*/
    } void rdb() {
        /*reads database into the right nodes and edges*/
    } public void paintNode() {
        /*draws and paints the nodes*/
    } public void paintEdge() {
    /*draws the edges and paints them*/
    } public void mousePressed() {
    /*listens for mouse click event*/
    } double weight() {
    /*calculates for greater edge weights*/
    } void append_pre_s() {
        /*this adds starting nodes from where process starts*/
    } void remove_pre_s() {
        /*this removes the starting nodes from the array*/
    } void findPath() {
        step1()  /*find the shortest path*/
        step2()
        step3()
        step3() }
```

RESULT ANALYSIS AND PERFORMANCE

When the program is running he circles with numbers in them are the nodes or stations or routers. The lines joining the nodes are the links/ linkages which are considered as distances between nodes (cables). The numbers on the linkages are the weights of linkages. The total cost of going from a starting node to a target node is written just around the targeted node. But before the movement starts, it must first listen for the mouse click. At the click of the mouse it goes from the start node assumed to be node 0 to node 1. It first paints the targeted node red and waits to hear the next click before changing the color to blue. The blue color indicates that the node has been visited. So it’s permanent. We also have the pleasure of changing the circle nodes to rectangles.

It is also pertinent to understand that Dijkstra routing algorithm is a global routing algorithm. This means that each node has the complete router intelligence of the entire network. The algorithm surveys the entire network before it can take decision on which route to take. All the network attributes are considered before routes are chosen.

Also notice that a blue line is drawn to trace the shortest path. Below is written the entire analysis of the result.

STEP 1:

The simulation appears like this fig.4.1a below waiting for an event to be taken by the user. When the mouse is clicked node 1 which is the closest to the node 0 changes to red. As you can see in fig.4.1b. You notice that the weight is turned to blue.

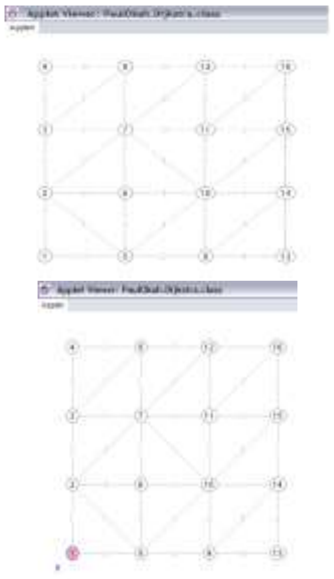


Fig.4.1 (a) & Fig.4.1 (b): Dijkstra simulations

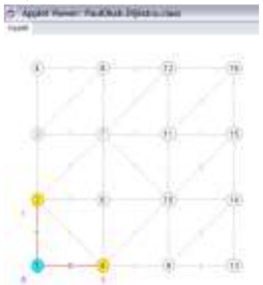
Arc(1)node (1) =0;

STEP 2:

When the mouse is clicked notice what happened. Node 1 turns blue and considers the yellow painted nodes. The yellow nodes that come under consideration are listed below;

Arc(2)node(2)=1;◄

Arc(2)node(5)=3;



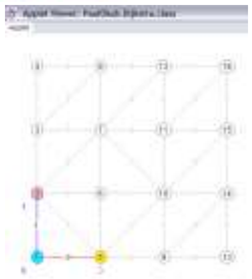


Fig.4.2 (a) & Fig.4.2 (b): Dijkstra Simulations

Notice that node 1 has turned blue. When you click the mouse it compares the weights of the yellow painted nodes. This is to decide which has the least weight. Once it does this, node 2 is painted red, showing it is the node with of least weight. This is therefore the shorter path of the two. Presented here are diagram of the simulations.

STEP 3:

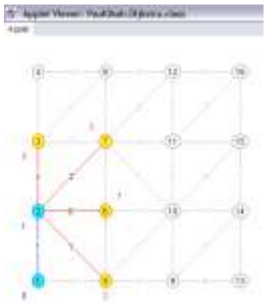
When an event is taking, the node 2 changes to blue with many others turning to yellow nodes. These are the neighbors under consideration. These nodes are listed below;

Arc(3)node(3)=1+2=3;

Arc(3)node(7)=1+2=3;

Arc(3)node(6)=1+6=7;

Arc(3)node(5)=1+1=2;◄



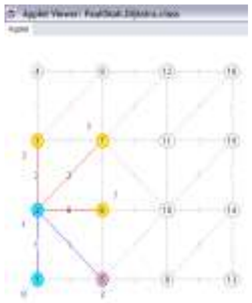


Fig.4.3 (a) & Fig.4.3 (b): Dijkstra Simulations

On an event occurring, the program runs a check on the least of nodes under consideration and decides which nodes to take. After the check it goes through route Arc(3)node(5) and paints it red. This shows that it is the shortest path to its destination. Below is the simulation diagram.

STEP 4:

When you click the mouse, the node 5 turns to blue color and start to consider several other nodes. This it shows by painting these nodes yellow. They are listed here along with there weights.

Arc(4)node(3)=1+2=3;◄

Arc(4)node(6)=1+1+3=5;

Arc(4)node(9)=1+1+2=4;

Arc(4)node(10)=1+1+1=3;

Arc(4)node(7)=1+2=3;

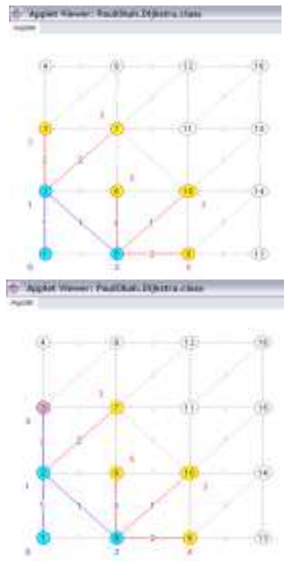


Fig.4.4 (a) & Fig.4.4 (b): Dijkstra Simulations

It runs a check to determine the shortest path, this it does by comparing the link weight. After which it paints Arc(4)node(3) red. This shows that it is the shortest path. The arrow indicates the chosen linkage.

STEP 5:

Clicking the mouse, the node 3 turns blue and several nodes come under consideration. These nodes are;

Arc(5)node(4)→1+2+3=6;
Arc(5)node(8)→1+2+1=4;
Arc(5)node(7)→1+2=3;◀
Arc(5)node(6)→1+1+3=5;
Arc(5)node(9)→1+1+2=4;
Arc(5)node(10)→1+1+1=3;

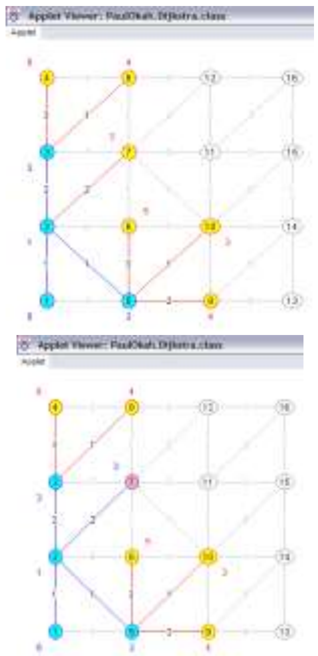


Fig.4.5 (a) & Fig.4.5 (b): Dijkstra Simulation

The node 7 is painted red showing that it is the shortest path.

STEP 6:

The node 7 is turned blue alongside its weight value and considers other neighboring nodes which are listed below.

Arc(6)node(4)=1+2+3=6;
Arc(6)node(8)=1+2+1=4;
Arc(6)node(12)=1+2+2=5;
Arc(6)node(11)=1+2+3=6;
Arc(6)node(6)=1+2+1=4;
Arc(6)node(10)=1+1+1=3;◀
Arc(6)node(9)=1+1+2=4;

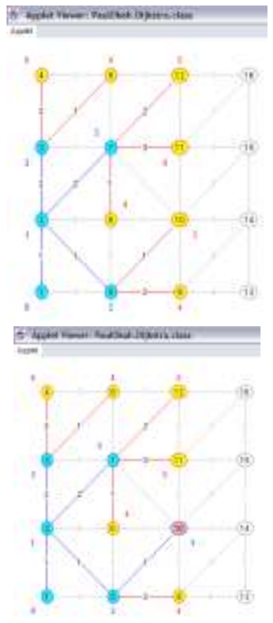


Fig.4.6 (a) & Fig.4.6 (b): Dijkstra simulations

The sequence continues like others.

STEP 7:

The node 10 turns blue with its weight value and its link while considering other nodes that are painted yellow.

Arc(7)node(4)=1+2+3=6;
Arc(7)node(12)=1+2+2=5;
Arc(7)node(11)=1+1+1+2=5;
Arc(7)node(15)=1+1+1+5=8;
Arc(7)node(14)=1+1+1+1=4;
Arc(7)node(6)=1+2+1=4;◀
Arc(7)node(9)=1+1+2=4;

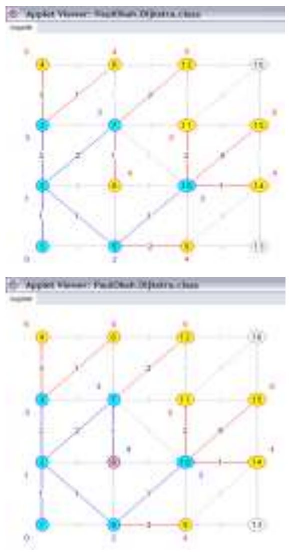


Fig.4.7 (a) & Fig.4.7 (b): Dijkstra Simulations

Notice that when the weights are the same it chooses any.

STEP 8:

The node 6 turns blue alongside its weight and link as it considers other yellow painted nodes. The list is here below;

Arc(8)node(4)=1+2+3=6; Arc(8)node(8)=1+2+1=4;
Arc(8)node(12)=1+2+2=5;
Arc(8)node(11)=1+1+1+2=5;
Arc(8)node(15)=1+1+1+5=8;
Arc(8)node(14)=1+1+1+1=4;
Arc(8)node(9)=1+1+2=4; ◄

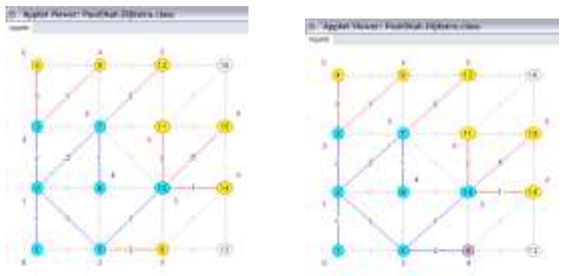


Fig.4.8 (a) & Fig.4.8 (b): Dijkstra Simulations

STEP 9:

The node 9 turns blue and starts to consider the neighboring yellow painted nodes. The list of these yellow painted nodes is here;

Arc(9)node(4)=1+2+3=6; Arc(9)node(8)=1+2+1=4; ◄
Arc(9)node(12)=1+2+2=5;
Arc(10)node(11)=1+1+1+2=5;
Arc(9)node(15)=1+1+1+5=8;
Arc(9)node(14)=1+1+1+1=4;
Arc(9)node(13)=1+1+2+4=8;

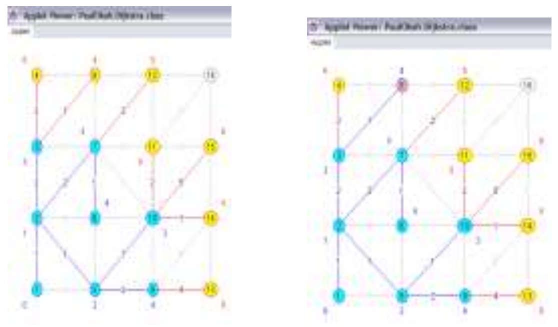


Fig.4.9 (a) & Fig.4.9 (b): Dijkstra Simulations

STEP 10:

The node 8 turns blue an begins to consider other neighboring nodes.

Arc(10)node(4)=1+2+3=6;
Arc(10)node(12)=1+2+2=5;
Arc(10)node(11)=1+1+1+2=5
Arc(10)node(15)=1+1+1+5=8;
Arc(10)node(14)=1+1+1+1=4; ◄
Arc(10)node(13)=1+1+2+4=8;

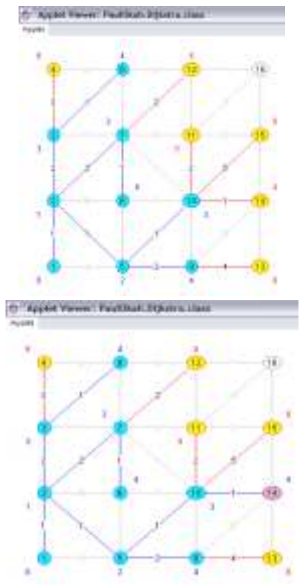


Fig.4.10 (a) & Fig.4.10 (b): Dijkstra Simulations

STEP 11:

The node 14 turns blue showing that it is the shortest path. The program begins to take a look at the surrounding nodes to determine the link with the least weight.

Arc(11)node(4)=1+2+3=6;
Arc(11)node(12)=1+2+2=5;
Arc(11)node(11)=1+1+1+2=5; ◄
Arc(11)node(15)=1+1+1+5=8;
Arc(11)node(13)=1+1+2+4=8;

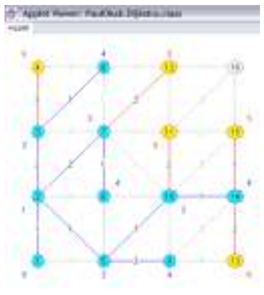
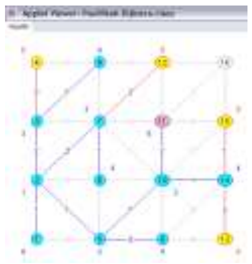



Fig.4.11 (a) & Fig.4.11 (b): Dijkstra Simulations

STEP 12:

The node 11 turns blue and paints the nodes under consideration yellow. They are;

Arc(12)node(4)=1+2+3=6;

Arc(12)node(12)=1+2+2=5; ◄

Arc(12)node(16)=1+1+1+2+3=8;

Arc(12)node(15)=1+1+1+1+1=5;

Arc(12)node(13)=1+1+1+1+2=6;

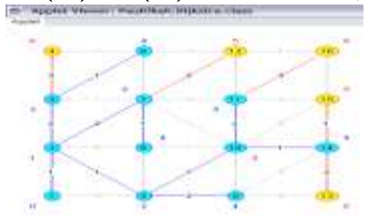
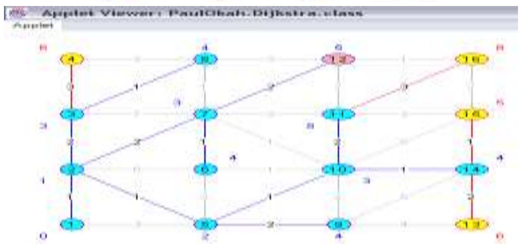



Fig.4.12 (a) & Fig.4.12 (b): Dijkstra Simulations

STEP 13:

The node 12 changes to blue and starts to consider other surrounding nodes. As usual it is to check the shortest path.

Arc(13)node(4)=1+2+3=6;

Arc(13)node(16
)=1++2+2+1=6;

Arc(13)node(15)=1+1+1+1+1=5; ◄

Arc(13)node(13)=1+1+1+1+2=6;

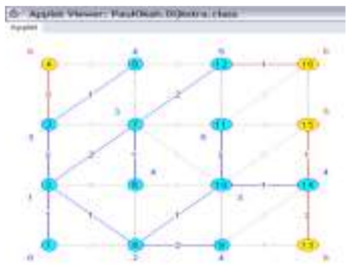
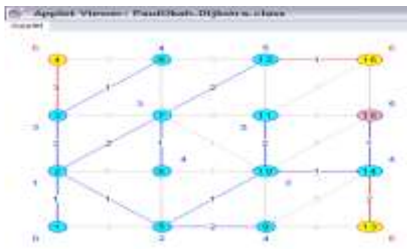



Fig.4.13 (a) & Fig.4.13 (b): Dijkstra Simulations

STEP 14:

The node 15 turns to blue and turns the neighboring nodes under consideration yellow. Notice that the link and the weight also turn blue.

Arc(14)node(4)=1+2+3=6; ◄

Arc(14)node(16)=1++2+2+1=6;

Arc(14)node(13)=1+1+1+1+2=6;

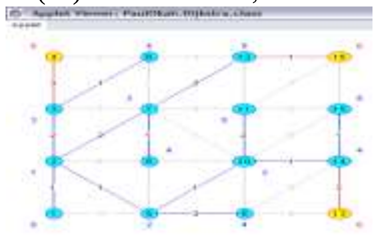
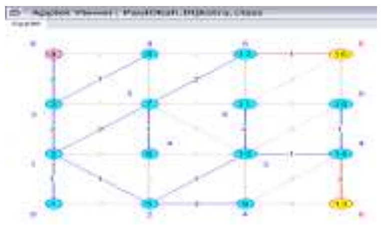



Fig.4.14 (a) & Fig.4.14 (b): Dijkstra Simulations

STEP 15:

The node 4 turns blue and starts to consider the neighboring nodes for decision taking.

Arc(15)node(16)=1++2+2+1=6;

Arc(15)node(13)=1+1+1+1+2=6; ◄

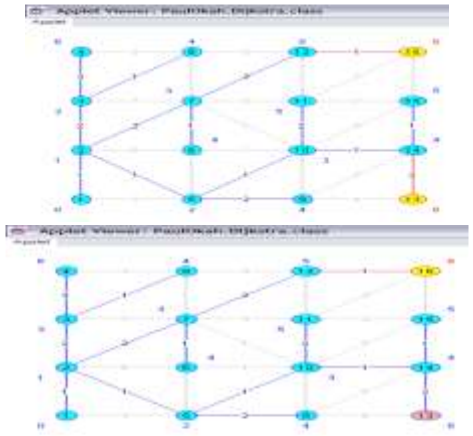


Fig.4.15 (a) & Fig.4.15 (b): Dijkstra Simulations

STEP 16:

The node 13 turns blue and considers the only remaining node.

Arc(16)node(16)=1++2+2+1=6; ◄

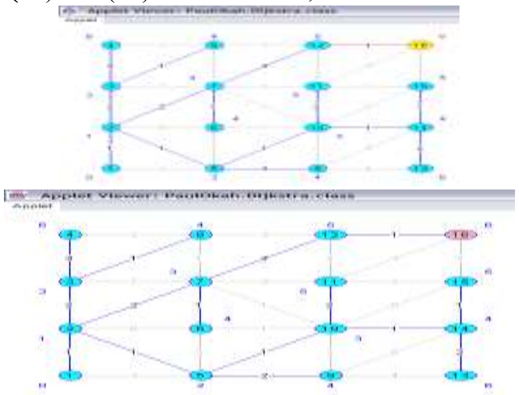


Fig.4.16 (a) & Fig.4.16 (b): Dijkstra Simulations

STEP 17:

The node 16 turns blue.

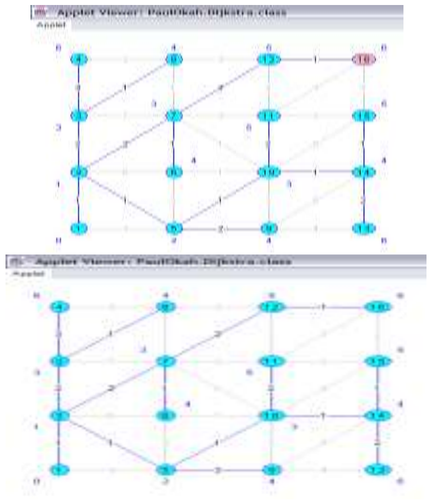


Fig.4.17 (a) & Fig.4.17 (b): Dijkstra Simulations

All the network stations or nodes have been visited as result all the nodes turn blue. We can choose any node in the network and start the simulation all over again.

The graphical representation of this Dijkstra algorithm is given at the appendix using the matlab codes. The result of this report shows that this algorithm stands load on the network by looking for the alternate route to a particular destination. It takes into consideration the congestion on the network. The performance is powerful.

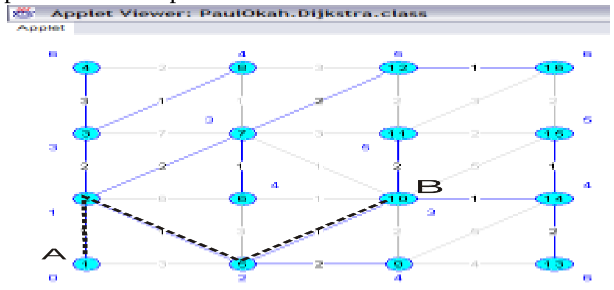


Fig.4.18 (a): Shortest path from A to B

Using A as the source node and B as the targeted node, the shortest path between them is shown by the thick dotted line. Calculating the cost is shown below:

node(1)=1, node(5)=1, node(10)=1;

totalCost=1+1+1=3;

Checking all other links to the same target will confirm this.

The illustration below shows a source node at C and the destination node at D. the total cost of weight of linkage i.e. the shortest path is calculated thus;

node(1)=1, node(2)=2, node(7)=2, node(12)=1;

totalCost=1+2+2+1=6;

going through several linkages to the target node will waist time. This can be proven by running a check on the simulated network.

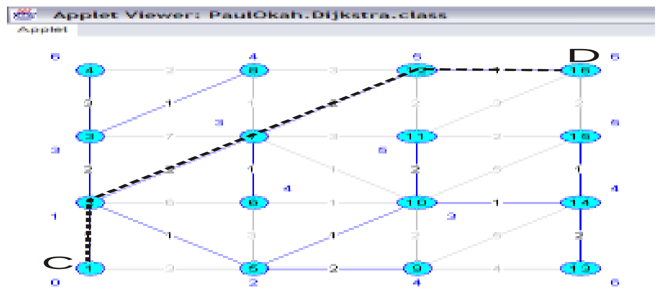


Fig.4.19 (a): Shortest path from C to D

4.2 Comparison of Results

E.W. Dijkstra came up with the algorithm in 1959. His algorithm is a global type routing algorithm. As a result each node has the capacity to survey the entire network before a decision is made on the shortest path to route information. It picks a starting node and from there picks any route and follows it till the end. If there is link failure, the entire network is redrawn and a new route picked. If it goes through to the end, it stores the cost or weight in an array. It comes back and picks another route. This will go on until all the routes are covered. It will now finally compare to determine which route is most efficient. Going by this little analysis, it is hereby shown that with the Dijkstra algorithm it will take a lot of time to route datagram from source to destination. Consequently, time is of great essence in communication.

The improvement that was done in this work is by the use of what is known as priority queue. This enables each node to actually prioritize some nodes. This it does by considering the nearest neighboring nodes to find the one of least cost. With each movement to different node comes this comparison. That is exactly what this work is doing. It continues this until it goes through to the end. Where there is a link failure it assigns infinity to the link weight and finds the nearest shortest path to its destination. It does not redraw the entire network and does not pick route one after another. The efficiency of this improved algorithm is much better as the time of routing is greatly reduced to barest minimum.

The Dijkstra routing algorithm does not make use of the priority queue system. This system was introduced in this research to improve the working of this novel Dijkstra routing algorithm.

CONCLUSION AND RECOMMENDATION

In today’s world, networking plays an important role in communication between autonomous computers. For this many hardware devices and software algorithms have been designed. So far, the traditional system used for the communication were the hub networking system and many other hardware applications present in the market, but as they operate on electricity, it may lead to the failure of device due to some malfunctioning in the hardware circuitry. Dijkstra Algorithm has been incorporated in ‘java’, which provides easy understandability and hence its chances of failure is negligible.. However, we have implemented the Dijkstra Algorithm in such a way that under the context of link failure it will provide the next shortest path from the available alternate paths that has been calculated along with the Optimum Path itself. As a result there is no need to execute the algorithm again.

Adaptive routing with constantly updated information is helpful in avoiding congested routes. Quality of service, quantity of service and speed are the three most important performance measures for any routing algorithm.

REFERENCES

Scheduling *algorithms* for multiprogramming in a hard real time environment .... and access control in fast packet-switched networks - *Guerin, Gun - 1992.*

Partridge, Craig, *1992* "A proposed flow Specification". RFC 1363. Pinto. A., Valente, R., Aguilar, R., Oliveira. J.L., 2000. "Managing *QoS* over IP". Internal Report.

D. yates J. Kuros , D Towaley and M. Hlochyl. “on per-to per session End to end delay distribution and the call admission problem for real time application with Qos requirements; in proceeding of SIGCOMM 93, 1993

Zheng wang and jon crowscoft routing algorithm for supporting resources reservation, dept of computer science, university college London, 1992

S, shenker D clark, L zhang A service model for an integrated services internet; internet draft, available for anonymous ftp at ds.intermic.net; internet-drafts/daft-shenker-realtime-model.00.txt October 1993

Richard Bellman; on a routing problem, quarterly of applied mathematics 16, 1958 william Byrd Press , INC, Richmond , VIrgina

R Guerin and L Gun A unified Approach to Bandwith Allocation And access control in fast packet-switch networks 1992

wallpaper April 2016 Oxford Cotswolds Guernsey Channel Islands Thames ...

Prof. Andrews' Publications - The University of Texas at Austin

users.ece.utexas.edu/~jandrews/publications.php

I no longer maintain all my *publications* on this webpage. Please ... J. G. *Andrews*, "A Tractable Framework for Coverage and Outage in Heterogeneous Cellular ...

Publications - University of St Andrews

https://www.st-andrews.ac.uk/intrel/publications/

University of St *Andrews* crest on white background ... The School of International Relations produces a variety of *publications* which promote the research of ...

Ilo S.F, Igbajar Abraham, *Enhancing network connectivity and Data security in Institutions with Remote Campuses, Using Virtual Private Network;* International Journal of Advanced and Innovative Research (2278-7844) / # 91 / Volume 4 Issue 10 2015

L zhang s s deering D Estrin S shenker and D zappala , a new resource reser vation protocol sept 1993

Patridge c , A proposed flow specification RFC RFC Jully 1992

Nwachukwu-Nwokeafor K.C, Igbajar Abraham, *Design of a Secured Online Voting System for electoral Process;* IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 12, December 2015. www.ijiset.com, ISSN 2348 – 7968,

Alberto Leon-Garcia and Indra Widjaja communication networks, Fundamental concepts and key architectures,.

Dijstra , E. W , (1959) " A note on two problems in connexion with Graphs" Mathemarik, vol 1, pp269-271.

Darren L, Spohn- data networks Design (2nd Edition), 1993,

D.P. bertsekas, optimal routing and flow control methods for communication networks in analysis and optimization of systems, A . bensoussan and J.L Lions, Eds, New york: Springre-Verlag, 1982.pp615-643

D. Bertsekas, R, Gallager, "Gallager, "Data networks", prentice-hall , 1992.

Charles E, perkins Adhoc networrking Addison wesley, woo1, ISBN 0201-309

A, parekh A, A Generalised proccessor sharing approach to flow control in integrated services networks. PhD thesis, Laboratory for information and decision systems massachusetts institute of technology , 1992

Christian Huitema. Routing Internet, 2nd Edition , practice hall, upper saddle River, 2000

C perkins, E Belding-Royer and S. Das Adhoc on demand Distance Vector (AODV) Routing RFC 3561,IETF Network working group, july 2003, category Experimental.

A Demers, S Keshav, and S shenker, analysis and simulation of a fair queueing algorithm. In journal of intrnetworking; ; research and experience 1, pp 3-26, 1990.

Changyong zhang , A novel mathematical model for the unique shortest path routhing problem . Department of mathematics , university of califonia 2005.

C Hendrick . routing Information protocol RFC 1058, networking group June, 1998

Andrew S. Tancenbarum, computer Networks, Fourth Edition. Pentice Hall International, Upper saddle river, 19996. ISB 0-13-394248-1,4

H. schulzrime, j kurose and D, Towsley. Congession control for real time traffic 1990

J.M jaffie and f Moss, a responsive distributed routing algoritm for computer networks july 1982

D. Clark, S Shenker and L Zhang " supporting real-time applications in integrated servings packet Network, architecture and mechanism , 1992

George C, sacket & Christopher Y, Mertz- ATM and multiprotocol networking 1997

G. meyer and s sherry triggered extensions to RIP to support Demand Circuits, RFC 2091, IETF Network working group January 1997

Chris D Clark - Publications - ResearchGate

https://www.researchgate.net/profile/Chris_Clark3/publications

... and research. Connect, collaborate and discover scientific *publications*, jobs and conferences. All for free. ... Martin Margold · Chris R. Stokes · Chris *D. Clark*.

David Clark's Selected Publications - UCL Computer Science

www0.cs.ucl.ac.uk/staff/D.Clark/pubs/

M. Boreale, *D. Clark*, and D. Gorla A Semiring-based Trace Semantics for Processes with Applications to Information Leakage. Mathematical Structures in ..

D, Clack and V Jacobon, Flexible and Efficient Resource management for Datagram Networks, presentation slides 1991.