International Journal of Soft Computing 3 (1): 13-23, 2008
ISSN: 1816-9503

Control of Traffic Congestion in a Packet-Switching Network Using Routing Algorithms

O.A. Afolabi, E.O. Justice and O.A. Isola
Department of Computer Science and Engineering,
Ladoke Akintola University of Technology, Ogbomoso, Nigeria

Abstract: This research discussed the application of routing algorithms to the control of the congestion of the traffic of packets in a packet-switching network, it is required to find the path with the least total cost of transmitting the packets, with the shortest possible distance between a packet source and a destination host and also to reduce the amount of time that takes the packet to get through to its destination. The routing algorithms considered are Dijkstra algorithm and the Floyd's algorithm, they are reviewed, implemented and compared with respect to there performances. The Floyd's algorithm implementation tends to find the path between all pairs of nodes in the network according to the changes in the costs of each path due to the packet passing through them. This makes Floyd's algorithm implementation one of the best ways of find paths through a network compared to the Dijkstra algorithm implementation which is good particularly for path finding between two specific nodes in the network.

Key words: Traffic congestion, packet, switching, routing algorithm, Floyd's algorithm, Dijkstra algorithm

INTRODUCTION

A packet-switched network is an interconnected set of networks that are joined by routers or switching routers. The main concept behind packet switched network is that any host (computers) connected to the network can send packet to other hosts on the same network or other network that is connected via the router. The packets sent within a network are units of data or information that is to be transmitted to a destination.

Packets contain header information that includes a destination address. Routers in the network read this address and forward packets along the most appropriate path to that destination. When there are a lot of packets sent through a particular path the weight or cost of the path increases therefore making transmitting more packet through the particular path slow, costly and time consuming. This consequently causes congestion in the transmitting of traffic of packets in this path and probably the whole network.

This cause of congestion prompts for a quick solution because it can affect the quality of service rendered by this kind of network. Routing algorithms in application to this traffic congestion control has been chosen for research purpose due to the graphical nature of the packet switched network. Routing algorithms are the stepwise procedures that are used by the routers to find the best route from a source host to a destination host. The algorithms work based on graphs, by scanning the graphs which represent the packet switched network to find the edges (path) with the lowest cost and with the shortest distance from any host (source) to any host (destination), therefore easing the effect of the congestion.

Based on how routers gather information about the structure of a network and their analysis of information to specify the best route, we have two major routing algorithms: Global routing algorithms and Decentralized routing algorithms. These algorithms are also known as DV (Distance Vector) algorithm and LS (Link State) algorithm, respectively. The link state algorithms make use of the Dijkstra shortest path algorithm to perform the function of its any-to-any path finding. The distance vector algorithms make use of the Bellman-Ford algorithms to perform its functions. There also some other algorithms that can be used, for example the Floyd algorithm.

Packets and packet switching networks: A packet can be described as a unit of data that is transmitted across a packet-switched network. A packet-switched network is an interconnected set of networks that are joined by routers or switching routers. The most common packet-switching technology is TCP/IP and the Internet is the largest packet-switched network. Other packet-switched network technologies include X.25 and IPX/SPX (the original Novell NetWare protocols).

Corresponding Author: O.A. Afolabi, Department of Computer Science and Engineering,
Ladoke Akintola University of Technology, Ogbomoso, Nigeria

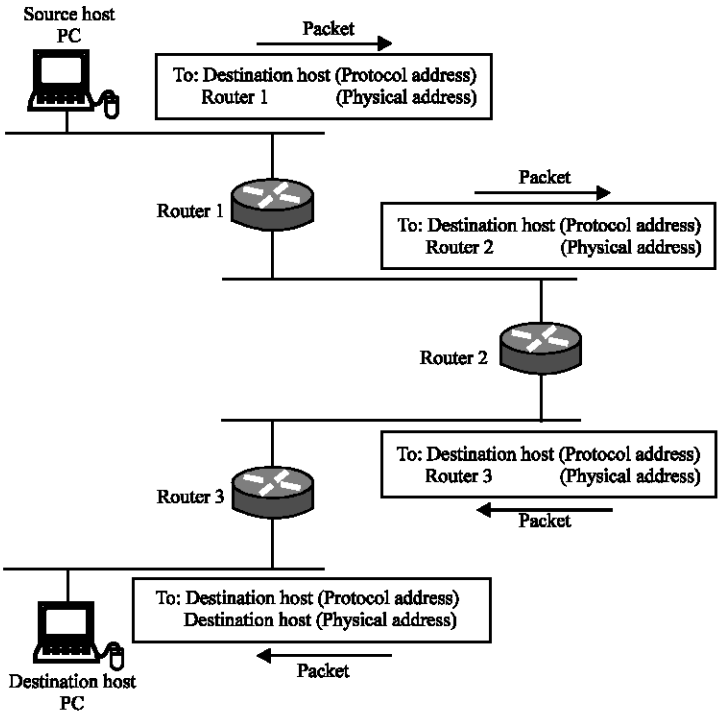


Fig. 1: Movement of packets within a network

The concept of a packet-switched network is that any host connecting to the network can send packets to any other hosts. The network is said to provide any-to-any service. The network typically consists of multiple paths to a destination that provide redundancy. Packets contain header information that includes a destination address. Routers in the network read this address and forward packets along the most appropriate path to that destination. In a simple packet-switched network, if computer A needs to send a packet to computer Z, the packet first travels to R1. By looking in a routing table, R1 determines that the best path to the destination is through the R2 interface. The network has a redundant topology; so, if the link between R1 and R2 fails, a path through R3 and R4 will reach R2 (Fig. 1).

Traffic congestion in a network: Congestion is a problem that occurs on shared networks when multiple users contend for access to the same resources (bandwidth, buffers and queues). Traffic congestion can also be thought of as a situation where by different packets are on a queue, to be transferred along a path of a very high cost in terms of bandwidth or a route that have already had enough packet and data been transferred through it causing the transfer rate of the link to be very slow.

In packet-switched networks, packets move in and out of the buffers and queues of switching devices through different paths as they traverse the network. In fact, a packet-switched network is often referred to as a "network of queues." A characteristic of packet-switched networks is that packets may arrive in bursts from one or more sources. Buffers help routers absorb bursts until they can catch up. If traffic is excessive, buffers fill up and new incoming packets are dropped. Increasing the size of the buffers is not a solution, because excessive buffer size can lead to excessive delay. Congestion typically occurs where multiple links feed into a single link, such as where internal LANs are connected to WAN links.

Congestion also occurs at routers in core networks where nodes are subjected to more traffic than they are designed to handle. Packets are injected by any host at any time and those packets are variable in size, which make predicting traffic patterns and providing guaranteed service impossible.

Congestion is caused because of several reasons:

- Buffer space scarcity causes congestion.
- Slow links lead to congestion.
- Slow processors cause congestion.

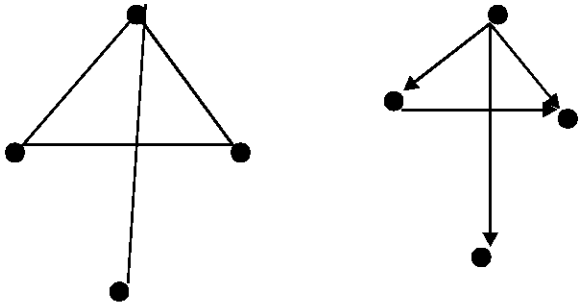
Traffic congestion control: The congestion of traffic of packets in a network as earlier said has a drastic effect on the operation and the quality of service that is delivered by the network. This particularly call for some appropriate congestion control mechanism to be put in place to ease the effect of the traffic congestion on the users of the

packet switching network which is ever busy. The data which are broken into packets by a process called fragmentation (the process of breaking a packet into smaller pieces so that they will fit into the frames of the underlying network). These fragments of packets are then transmitted from the source through different paths in a network to be reassembled at the destination. When packets are flooded on a single path the link becomes slow and continuous transmission of packets through this path causes it to get congested. The congestion control mechanism that is studied in this project is the use of routing algorithms (Dijkstra and Floyd algorithms) to find the shortest route to a destination by avoiding the links that is heavy with traffic which is represented in terms of the weight of the edges of the graph which is dissected by the algorithms. This make the data to reach its destination faster and thereby reducing the time delay due to the packet taking a congested path.

Routing algorithms: Routing is the act of moving information across an internetwork from a source to a destination. Along the way, at least one intermediate node typically is encountered. Routing involves two basic activities: Determining optimal routing path and transporting information groups (typically called packets) through an internetwork (Vadlamudi, 2005). The topic of routing has been covered in computer science literature for more than two decades, but routing achieved commercial popularity as late as the mid-1980s. The primary reason for this time lag is that networks in the 1970s were simple, homogeneous environments. Only relatively recently has large-scale internetworking become popular (Cisco systems, routing basics, 92-2002). In routing the nodes (routers) encountered uses what is called routing algorithms to find destinations.

Routing algorithms is a stepwise procedure, which the router utilizes in finding the shortest and the most efficient path for packets in situation of traffic congestion in which large numbers of different packets are to be sent from different sources to different destinations. Routing protocols are protocols that implements routing algorithms, they are used by intermediate system (routers) in determining paths. Routers run routing protocols to discover neighboring routers and the networks attached to them (Steve Brasher, 2004). These protocols let routers exchange information about the network topology as it changes due to new or failed links.

Basis of routing algorithms: The routing algorithms are generally based on graph theory (Roozbeth Razavi, 2004). In a brief overview of graph theory, a graph can be defined as a network of vertices (nodes) and edges that are connecting the vertices. There are different sorts of graphs which are classified according to the nature of the edges that are connecting the nodes of the graph. For example we have simple graphs (in which at most one edge is connecting two vertices), multi-graphs (in which multiple edges are allowed between the vertices). Also, a graph edge can be weighted and the vertices may be labeled depending on the operations performed on both of them.



Simple graph **Directed graph**

Here are some terminologies that are used in a graph:

Path: A path a set edges that is joining two nodes, there are also sets of intermediate nodes in the path.

Connected graph: This is a graph G which is said to a be connected if there is a path in G joining any two nodes of the graph.

Tree: A tree is a connected graph, which has no loop.

A network: It is a graph such that flows can the place in the branches of the graph.

A graph is a perfect model of the packet switching networks which is been considered in this project.

There are different types of routing algorithms, for example Dijkstra algorithm, the Bellman-Ford's algorithm, Floyd's algorithm, Dantzig shortest path algorithm just to mention a few of them. They are used in the routing protocol used by the routers in the packet-switching network to find the shortest and the most efficient route for prompt data deliver (Jun Wang and Kalara, 2001). The routing algorithms that are in this work are the Dijkstra algorithm and the Floyd's algorithm.

Dijkstra algorithm: Dijkstra algorithm determines the distances or weight (costs) between a given vertex and all other vertices in a graph. Dijkstra algorithm creates labels associated with vertices. These labels represent the distance or weight (cost) from the source vertex to that particular vertex. Within the graph, there exist two kinds

of labels: Temporary and permanent. The temporary labels are given to vertices that have not been reached. The value given to these temporary labels can vary. Permanent labels are given to vertices that have been reached and their weight (cost) to the source vertex is known. The value given to these labels is the distance or weight (cost) of that vertex to the source vertex. For any given vertex, there must be a permanent label or a temporary label, but not both. The algorithm begins by initializing any vertex in the graph (vertex A, for example) a permanent label with the value of 0 and all other vertices a temporary label with the value of 0. The algorithm then proceeds to select the least cost edge connecting a vertex with a permanent label (currently vertex A) to a vertex with a temporary label. Vertex B's label is then updated from a temporary to a permanent label. Vertex B's value is then determined by the addition of the cost of the edge with vertex A's value. The next step is to find the next least cost edge extending to a vertex with a temporary label from either vertex A or vertex B (vertex C, for example), change vertex C's label to permanent and determine its distance to vertex A. This process is repeated until the labels of all vertices in the graph are permanent.

When mathematically expressed Dijkstra algorithm can be explicitly explained as follows: The algorithm works by keeping for each vertex v the cost $d[v]$ of the shortest path found so far. Initially, this value is 0 for the source vertex s and infinity for all other vertices, representing the fact that we do not know any path leading to those vertices. When the algorithm finishes, $d[v]$ will be the cost of the shortest path from s to v or infinity, if no such path exists.

The basic operation of Dijkstra algorithm is edge relaxation: If there is an edge from u to v, then the shortest known path from s to u can be extended to a path from s to v by adding edge (u, v) at the end. This path will have length d[u] +w (u, v). If this is less than d[v], we can replace the current value of d[v] with the new value.

Edge relaxation is applied until all values d[v] represent the cost of the shortest path from s to v. The algorithm is organized so that each edge (u, v) is relaxed only once, when d[u] has reached its final value.

The algorithm maintains two sets of vertices S and Q. Set S contains all vertices for which we know that the value d[v] is already the cost of the shortest path and set Q contains all other vertices. Set S starts empty and in each step one vertex is moved from Q to S. This vertex is chosen as the vertex with lowest value of d[u]. When a vertex u is moved to S, the algorithm relaxes every outgoing edge (u, v).

In the following algorithm, u: = Extract-Min (Q) searches for the vertex u in the vertex set Q that has the least d[u] value. That vertex is removed from the set Q and then returned.

To find the best route in a network, the routers make use of algorithms such as the Dijkstra shortest path algorithm. In this algorithm, a router, based on information that has been collected from other routers (which are initially obtained by the routers by the transmission of hello packet at start up to know the situation of the network), builds a graph of the network. This graph shows the location of routers in the network and their links to each other. Every link is labeled with a number called the weight or cost. This number is a function of delay time, average traffic and sometimes simply the number of hops between nodes. For example, if there are two links between a node and a destination, the router chooses the link with the lowest weight.

The Dijkstra algorithm goes through these steps:

- The router builds a graph of the network and identifies source and destination nodes, as V1 and V2 for example. Then it builds a matrix, called the "adjacency matrix." In this matrix, a coordinate indicates weight. For example, [i, j] is the weight of a link between Vi and Vj. If there is no direct link between Vi and Vj, this weight is identified as "infinity."
    - The router builds a status record set for every node on the network. The record contains three fields:
    - Predecessor field-the first field shows the previous node.
    - Length field-the second field shows the sum of the weights from the source to that node.
    - Label field-the last field shows the status of node. Each node can have one status mode: "Permanent" or "tentative."
- The router initializes the parameters of the status record set (for all nodes) and sets their length to "infinity" and their label to "tentative."
- The router sets a T-node. For example, if V1 is to be the source T-node, the router changes V1's label to "permanent." When a label changes to "permanent," it never changes again. A T-node is an agent and nothing more.
- The router updates the status record set for all tentative nodes that are directly linked to the source T-node.
- The router looks at all of the tentative nodes and chooses the one whose weight to V1 is lowest. That node is then the destination T-node.
- If this node is not V2 (the intended destination), the router goes back to step 5.
- If this node is V2, the router extracts its previous node from the status record set and does this until it arrives at V1. This list of nodes shows the best route from V1 to V2.

Floyd's algorithm: This is another standard algorithm that is used as routing algorithm. In Floyd's algorithm two arrays is set up, one for the set of distance of the edges between the nodes called the distance matrix and the other for the set of nodes in the graph or network. Floyd's Algorithm works by first creating a two-dimensional array that stores information about the presence or absence of a link between each node and every other node. In short, it is a map of the subnet in array form. Once this is done, it tries to compute the shortest path between the source and destination node (which are chosen by the user). The metric used here is the number of hops (the trip a packet takes from one router or intermediate point to another in the network). The algorithm tries to find a route with the minimum number of hops between the source and destination.

The algorithm starts off by parsing the array to check if a direct link is available between the source and destination nodes. If so, this is the "shortest path". If not, then it checks if a valid route can be obtained by placing any one intermediate node. The first valid route is returned to the user. If no route is found, then it checks again with all combinations of two intermediate nodes. If still no valid route exists, three intermediate nodes are used. And similarly, more and more intermediate routes are added until a valid route between the source and destination is found, which is then displayed to the user. Since the algorithm starts off with the minimum number of intermediate nodes and then adds more, obviously the first valid computed route will be the shortest and there is no further need to compute any alternative routes. The Floyd's algorithm is expressed below:

If a path leads from A to B and another path leads from B to C than there is a path from A to C.

- For (int A=0; A<V; ++A)
- For (int B=0; B<V; ++B)
- If (Adj [A] [B])
- For (int C=0; C<V; ++C)
- If (Adj [B] [C])
- Iif (! Adj [A] [C] || (Adj [A] [B] + Adj [B] [C] < Adj [A][C]))
- Adj [A] [C] =Adj [A] [B] +Adj [A] [C];

A graph will be generated after the number of nodes to be generated has been entered. In the above example the number of nodes entered 7, therefore the one above is to generate graph with 7 nodes as shown in Fig. 3.

The next step after generating the network is the path finding process therefore the start and destination nodes will be specified in the dialog boxes that pops up when the find path button is clicked (Fig. 4 and 5).

The dialog box to enter the start and destination node will then be entered then the Walked path will be outputted. In the below example the start is node 1 while the destination is node 4. The walked path is 1---2----4. This is 2 minimal paths from the source to the destination as shown in Fig. 6.

There are other paths that can be taken is this graph but the one chosen is the one with the lowest hops, that is hops with the minimum number of routes to the destination (Fig. 7).

This dijkstra algorithm implementation finds the shortest path that a packet can take from a source to a destination node by minimizing the number of intermediate nodes that it will pass through thereby avoiding any instance of encountering congestion.

THE FLOYD'S ALGORITHM IMPLEMENTATION

The Floyd's algorithm is divided into sections implemented with Visual Basic 6.0.

- There is a presumed graph that represents the network; on each paths of this graph is the cost (distance) of connecting each node in the network. The graph consists of 5 nodes and 9 paths as shown in Fig. 8.
- The problem of finding the shortest path between all pairs of vertices on a graph is akin to making a table of all of the distances between all pairs of nodes in the network. The Floyd's All-Pairs-Shortest-Path algorithm uses a dynamic-programming methodology to solve the All-Pairs-Shortest-Path problem.

```
FLOYD-WARSHALL(W)

n = rows(W)
D(0) = W
for k = 1 to n
    do for i = 1 to n
        do for j = 1 to n
            d_ij^(k) = min (d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D(n)
```

- The algorithm outlined above provides the shortest distance between all of the vertices in terms of a table (an adjacency matrix).
- The main implementation consists essentially of two sections. The left portion of the interface shows a pictorial representation of the graph that is specified in an adjacency matrix that appears to its right. (Note: To simplify the coding I have used dots rather than arrow heads to indicate the direction of a particular edge. Also two colors, black and red, have been used to distinguish forward and back edges (based purely on node number)).

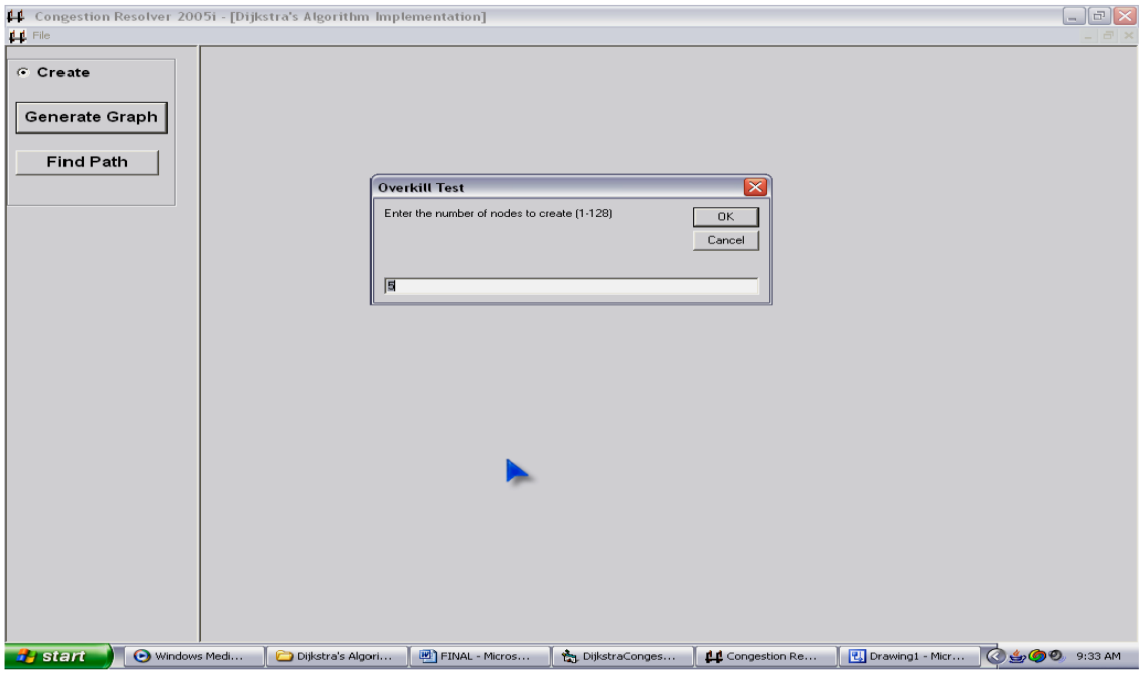


Fig. 2: To enter the number of nodes to be created

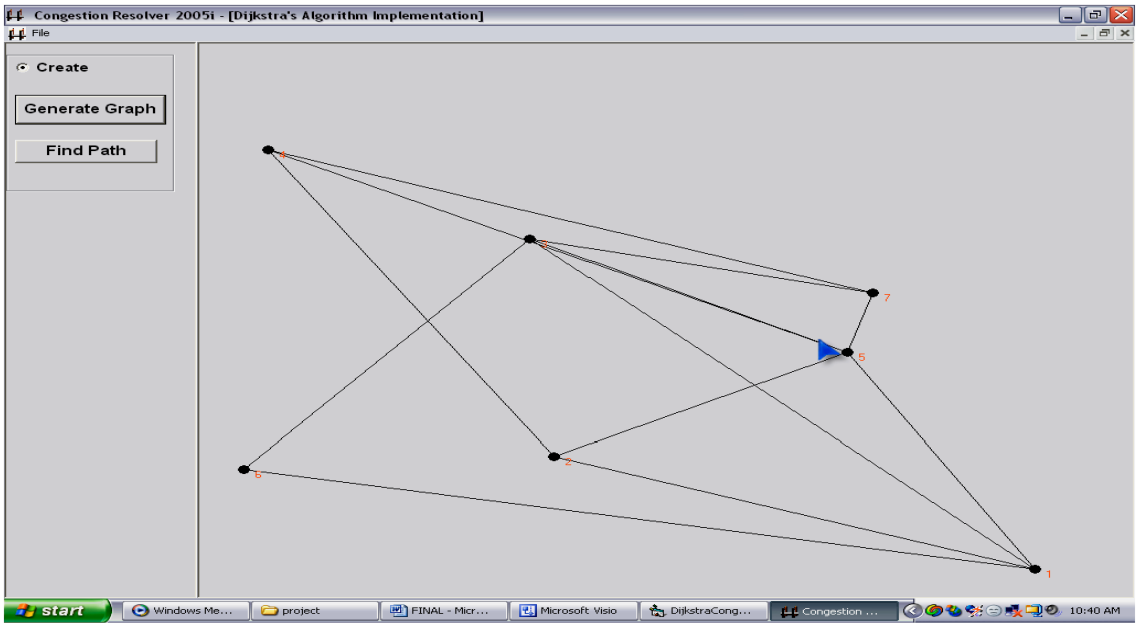


Fig. 3: Generated graph (network)

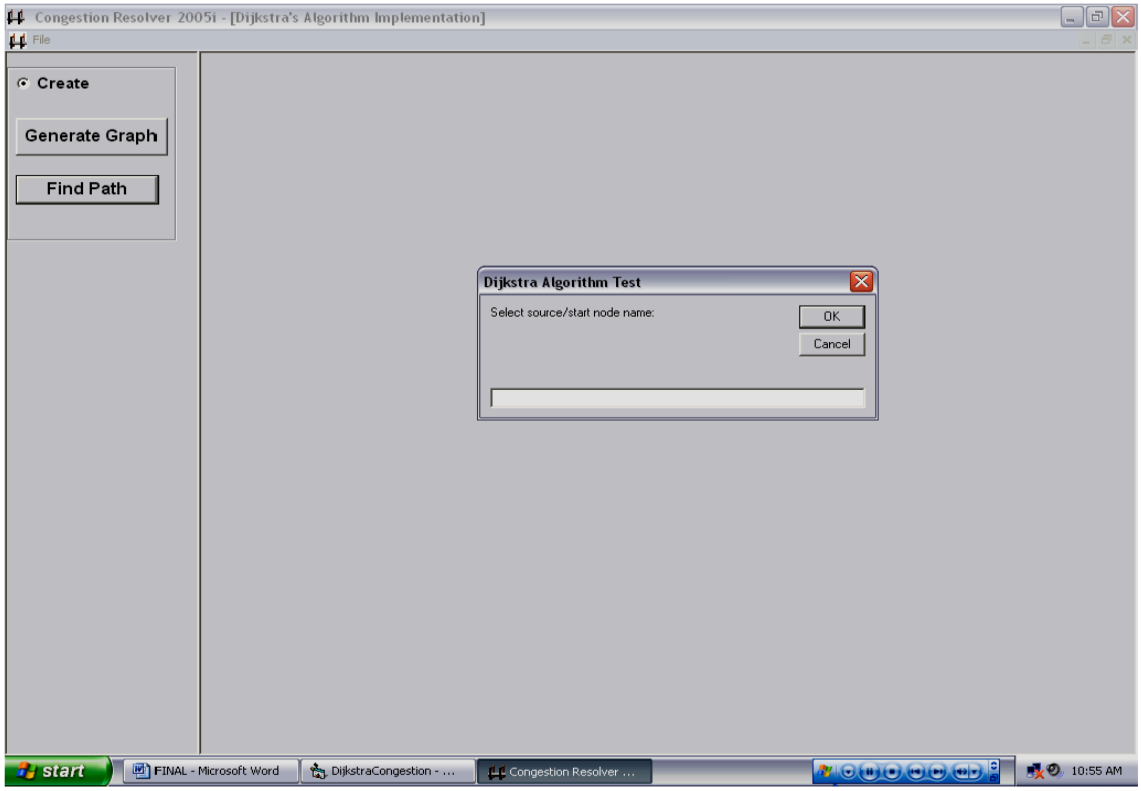


Fig. 4: Dijkstra algorithm test to initialize the start node

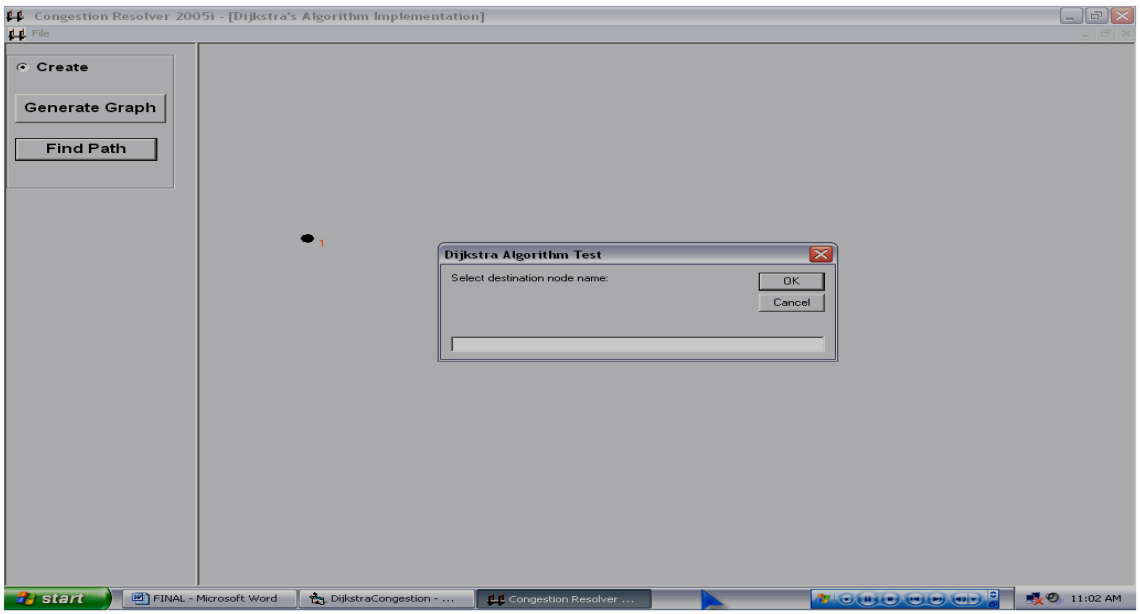


Fig. 5: Dijkstra Algorithm Test to initialize the destination node

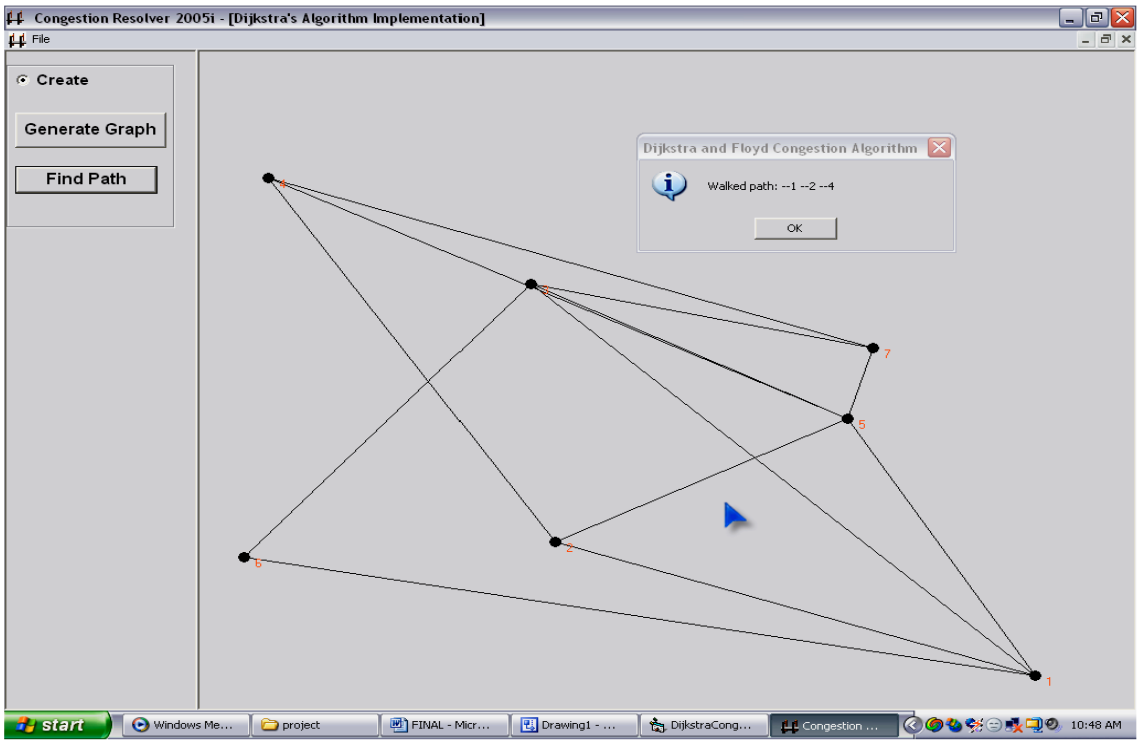


Fig. 6: Interface showing the walked path taken by the packet

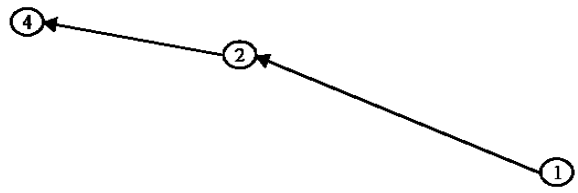


Fig. 7: The walked path

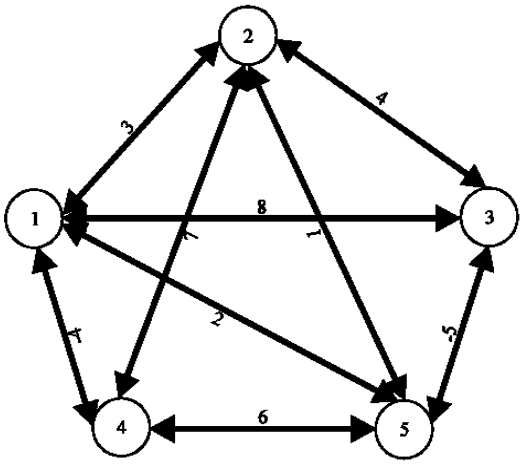


Fig. 8: The assumed network

- The adjacency matrix has been preloaded with the values that give rise to the graph depicted above. The weights appearing in the adjacency matrix can be altered by the user. In order to update the graph such that it corresponds to an altered matrix simple click on the 'Update' button. You can return to the default values by clicking on the 'Reset' button.
- The calculation of the shortest paths is initiated by selecting the 'Floyd-Warshall' button. During this calculation the adjacency matrix is replaced by the shortest path matrix and the values are updated (and highlighted) as the algorithm progresses. If the 'single step' checkbox has been marked then the application should pause at the end of each cycle. Once the full Floyd-Warshall calculation is complete, each entry in the matrix provides the shortest distance each pair of nodes. For example if a value of 1 lies at the intersection of the 5th row with the 3rd column than this is the shortest distance from node 5 to node 3. After the calculation is complete the actual shortest path between any two nodes is displayed by being superimposed on the graph itself. The 'choice box' enables the path between any two vertices to be chosen.

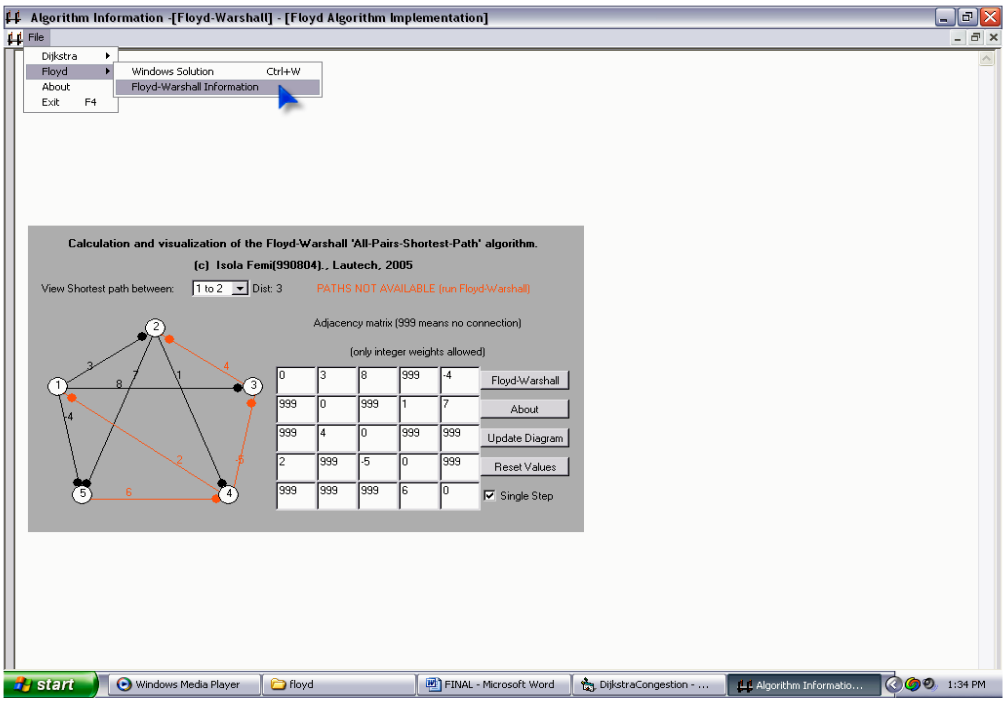


Fig. 9: The Floyd's algorithm implementation interface

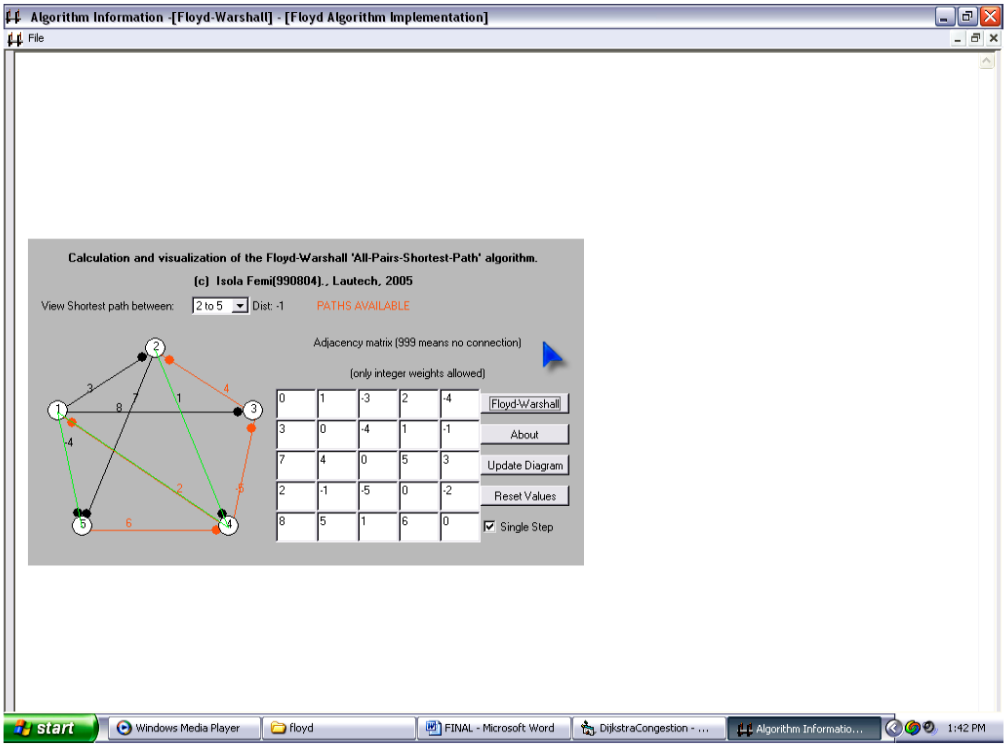


Fig. 10: Finding path using the Floyd's algorithm implementation

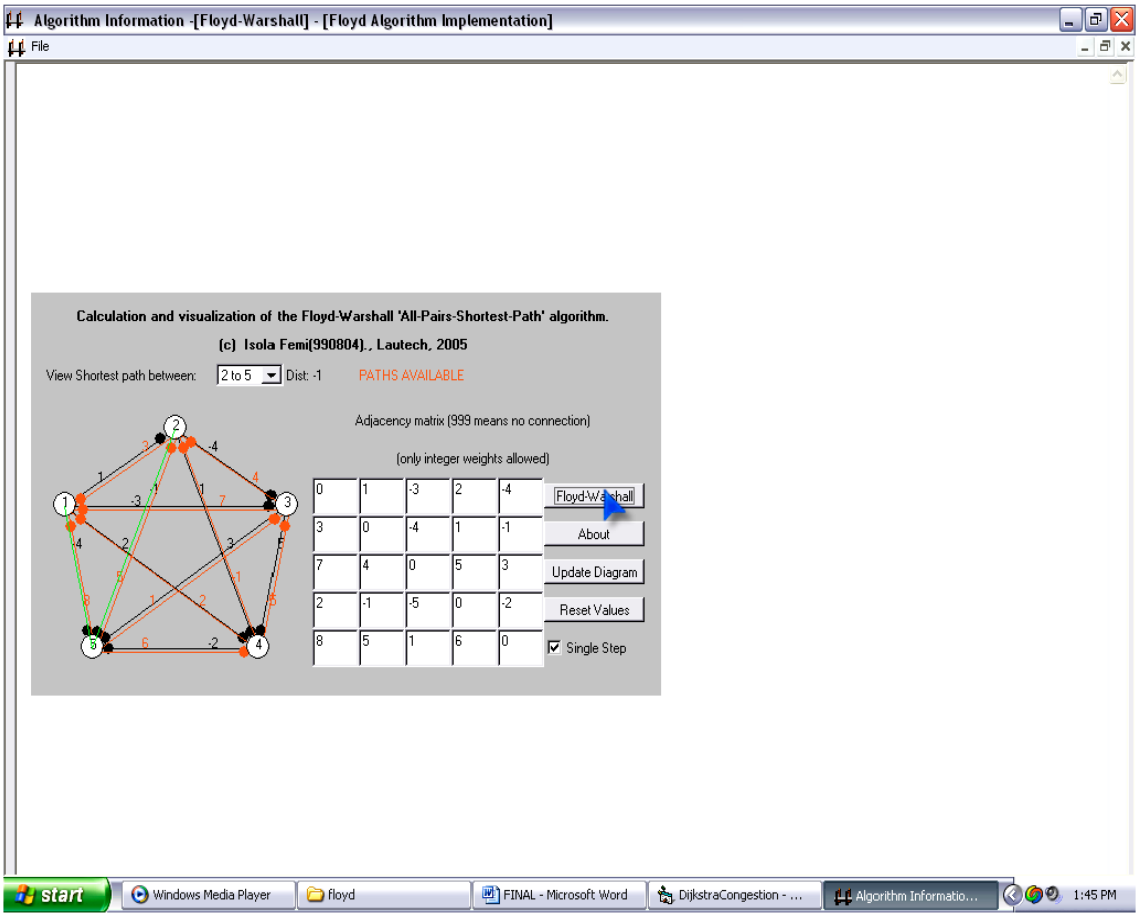


Fig. 11: Executing the Floyd's algorithm in steps

For example if we are to find the shortest path from node 2-5 using this Floyd's algorithm implementation according to Fig. 9. After entering the source and destination nodes the Floyd's algorithm will perform the following steps of its process. The first step of the process is in given Fig. 10.

The second and the final step will be:

This stepwise process is possible because of the checked single step box, the steps can be skipped top the final one by unchecking the single step box.

The path is shown by the green lines in the graph which represents the shortest possible path from a for a packet to get to destination node 5 from source node 2 using the Floyd's algorithm in finding the path from the source as in Fig. 11.

The cost of each path that is represented by the numbers on each path can be in terms of the distance between the two points on the network or it may be in terms of the rate of congestion of the connecting paths. This give the particular way in which the congestion is to be controlled, that is the congested path avoided, the lengthy distances avoided and the number of hops a packet have to make before it get to its destination. Also, it is the parameter with which the router (represented by the nodes) will find the path for each and every packet originating from it or arriving at it on its path to its destination according to the header messages (information about the where the packet is heading) attached to it. The Floyd's algorithm implementation tends to find the path between all pairs of nodes in the network according to the changes in the costs of each path due to the packet passing through them.

This makes Floyd's algorithm implementation one of the best ways of find paths through a network compared to the Dijkstra algorithm implementation which is good particularly for path finding between two specific nodes in the network.

CONCLUSION

The traffic congestion that frequently occur in a packet switching network can be controlled and its effect on the network minimized by making sure each and every packet transmitted on the network is effectively routed so that it will not increase the congestion of already

congested paths.As examined in this project the two algorithms if implemented properly will control congestion in the network effectively.

The Floyd's algorithm finds the path for the packets between any two nodes in the network which makes it a parallel algorithm which is the edge it have over Dijkstra algorithm. Dijkstra algorithm though it perform the same function as the Floyd's, it finds the path for the packet between a particular node and any other nodes in the network.

Therefore, it is concluded that the use of either of these routing algorithms will control traffic congestion in a packet switching network.

REFERENCES

Cisco Systems Inc., 2002. Routing Basics. http://www.cisco.com/ routing basics.html.

Dijkstra Algorithm, 2004. Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/ Dijkstra Algorithm.htm

Jun Wang and Klara Hahrstedt, 2001. Hop-by-Hop Routing Algorithms for Premium Traffic. (Published Thesis) University of Illinois, Urbana- Champaign, Urbana IL 601801, USA.

Roozbeth Razavi, 2004. How Routing Algorithms Works. http://computer.howstuffworks.com/Routing_Algoritims.htm

Steve Brasher, 2004. How Routers Works. http://computer.howstuffworks.com/ Routers.htm

Vadlamudi, S.R., 2005. Network Congestion Control. Unpublished semester project, The School of Technology, Indiana State University Terre Haute, Indiana.