International Journal of Engineering Research and General Science Volume 3, Issue 3, May-June, 2015
ISSN 2091-2730

Efficient Dynamic SPT algorithm for Network Routing

Snehal Kolte

M. E. (C.E.) Second Year

Department of computer engineering, DYPSOE,

Pune University, Pune, Maharashtra, India

Email: snehalkolte2010@gmail.com

Soumitra Das

H.O.D.

Department of computer engineering, DYPSOE,

Pune University, Pune, Maharashtra, India

Email: soumitra_das@yahoo.com

Abstract— Implementing a high speed routing is important in the network. Each router has a database maintained having whole network system topology in routing table. Each router should update its routing table rapidly if network topologies get modified to maintain high routing speed. There are several methods to achieve such functionality. Old classical method doesn't provide a satisfactory solution for rapidly changing networks topology. The fast construction of Shortest Path Tree (SPT) is important to achieve fast routing speed in a network. Whenever the network topology changes, the old shortest path tree must be updated fast. This paper presents improvements to existing landmark based shortest path estimation methods. The proposed dynamic algorithm constructs new SPT as network topologies get changed.in this dynamic algorithm, only edges which got weight changed and contributes to the construction of the new SPT will be considered and SPT is constructed dynamically..

Keywords— dynamic routing, shortest path, network routing, Shortest path Tree (SPT), Local Landmarks, Weighted Graph, Static Routing algorithms, Dynamic routing.

INTRODUCTION

Network Routing is a process which is choosing a way of sending communication data in a certain network. Network routing process is usually performed based on a routing table that manages various network destinations' routes. Therefore, the routing table formation written in a memory of the router is very important for effective routing. There are many graph algorithm methods used in routing algorithms. Each link is composed of a pair of Nodes. In a network routing, the nodes of graphs represent routers, and the links which connect these nodes represent physical links between routers.

In today's Internet, demands for broadband Internet Applications have grown rapidly. Therefore, high speed routing has become more important at Open Shortest Path First (OSPF) which is the most used intra-autonomous system routing protocols. When topological changes occur due to an unexpected situation at the OSPF, network routing algorithms are used to update the routing table. For example, if there is a link failure in a network, then the Shortest Paths have to be re-computed[13]. Normally in this case, the shortest paths computation is performed by re-running the algorithm. However, when links acquire new weights in a network, the SPT whole SPT gets updated which can increase the computation time and cause unnecessary corrections by repeating its operation in all of the nodes where the link's weight does not change.

However, this well-studied static algorithm becomes very inefficient when a small portion of the SPT needs to be updated in a network. This is because one link change results in computation of the full tree in each router and entries updating in its routing table [4],[5]. In many cases, the new SPT shows a little modification compared with the old SPT[11],[12] or no difference at all. The static algorithm for the SPT [6],[7] update is having a lot of unnecessary computing and routing table entry updates. Thus, it is very important that algorithms for dynamically updating SPT should get introduced to handle the link state changes in a network efficiently.

This happens because one link change results in re-computation of the whole tree in every router and entries updating in its routing table. The new SPT shows a little modification compared with the old SPT or no difference at all. The method using the static

algorithm for the SPT update incurs a lot of unnecessary computing and routing table entry updates. Thus, it is very important that algorithms for dynamically updating SPT are introduced to handle the link state changes in a network efficiently. To get less computation time, the updating process to separate weight-increase operations and weigh- decrease operations is proposed for dynamic SPT [8],[9] update. In this paper, a new algorithm is proposed based on the analysis of the probability of edges used to the construction of the new SPT.

The number of edges considered in the new algorithm is far less than any other algorithms [8],[10]. The proposed algorithm not only reduces the computational complexity required to update an old SPT, but also maintains the routing table stability by keeping the topology of an old SPT.

EXAMPLE

In an example of Figure 1, A graph is shown, each node is labeled with letters (A to P), represents a routers. And the weight of one link between two nodes represents a link state cost means network traffic delay time between two routers. If an edge e is u → v, node u is the source node of the edge while node v is the end node. The number inside each node specifies the shortest distance from the source node A (tree node) based on the given graph.

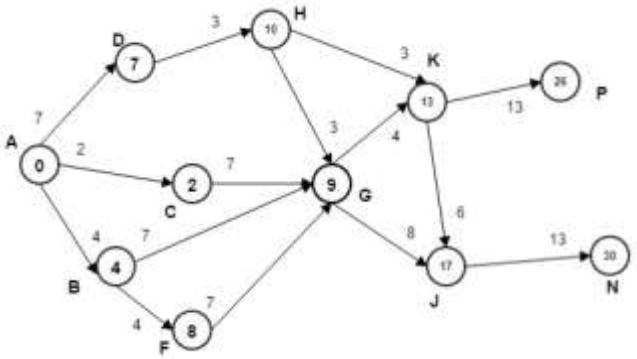

Figure.1. Network with their weight (delay Time)

Take a simple example where the weight of edge from node B to G decreases from 7 to 2, then the SPT needs to update. The shortest paths for nodes outside of the area encircled can be the same in both the old and new SPT. While for nodes inside the area within the circle, their SPT should be updated. These nodes are all child of node G (including G itself) following the old SPT. The new SPT is shown in Figure 2. We will figure out the underlying properties to dynamically generate the new SPT from the old one.

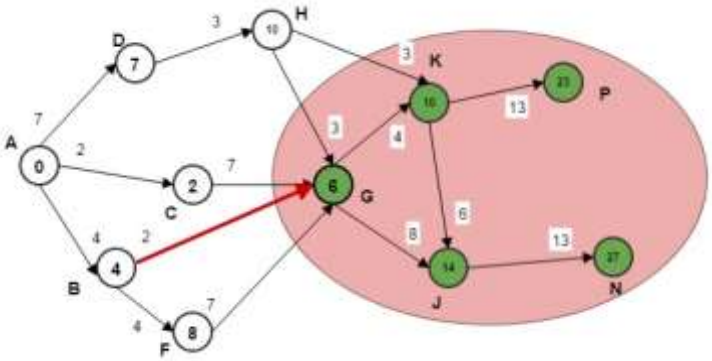

Figure.2. Network with their weight (delay Time)

For all nodes encircled, their shortest distances can be either decreased by 2 if their shortest paths follows the ways in the old SPT, or decreased not much (more than 2) by selecting some other paths through nodes outside of the circle. For the nodes inside the area encircled, i.e. G,J,K,N, P, there may exist several incoming edges to them. Each incoming edge can make its end node by a new distance from the tree node A. Only the edge with the smallest increment to a node, which needs to be updated, should be considered to satisfy the shortest distance property.

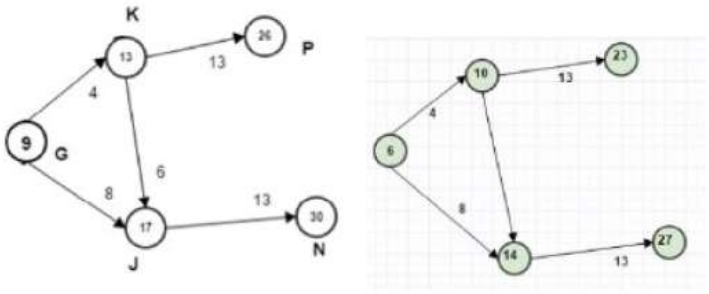

Figure.3. (a) Nodes with one incoming edge to denote the smallest increments; (b) The useful edges to construct the new SPT

For example, edge (C,G) and (B,G) can both reach node G. Through the former edge, the shortest distance to node G will be decreased. Thereby, edge (B,G) is meaningful for the updating process while edge (C,G) is not. We list all significant edges in Table 1.

Node	G	J	K	N	P
Incoming Edge	(B,G)	(F,J)	(G,K)	(J,N)	(K,P)
Decreased Value	3	3	3	3	3
Included or Not	Yes	Yes	Yes	Yes	Yes

Table.1. AN EXAMPLE OF A TABLE

In Table 1, nodes that should get updated are listed in the first row. The edge in the second row indicates the new path to its end node with the smallest decrement value compared with the old shortest distance. And the third row is the represents a decreased value.

For example, if the updated shortest distance path to node P goes through edge (K, P), the new distance should be 26-3 = 23. The Significant Edge is defined in this paper as the edge only in the new SPT (not in the old SPT).

All updated edges in Table 1 will get added to a queue Q. Q is an edge list that includes some edges with related information. The nodes to be updated and their related edges are shown in Figure 3(a). Every incoming edge has a cost to show the increment to the shortest distance of its end node if the shortest path through the edge. The number in the node is the smallest increased value among all its incoming edges. As we can see that only smaller portion of the SPT is getting updated as weight of the one node changes. Few have never been used, such as edges (M, P), (F, J) and (I, N). Thus, we do not need to put them into the edge set Q first and remove them later. If we only keep the incoming edge on condition that has increased its value is smaller than the ones of all its ancestor nodes, the new graph can be looked like Figure 3(b). The number in the node is the smallest one either from the decrements by its incoming edges, or from the value of its ancestors. The graph of changed weight can be easily calculated using Depth-First-Search algorithm, which is a comparison of its parent's value and the smallest increased value among its all incoming edges.

If we updates only the nodes which got weight changed, the computation time of SPT will be much lesser than that of computing a whole new SPT from old one.

DYNAMIC SPT ALGORITHM

Static routing algorithms should be applied when computing the shortest paths where some links have new weights near the root node. The reason that static routing algorithms are applied in this situation is because there are a lot of nodes which have to be computed near the root node. In this case, using static routing algorithms is a better method to compute the shortest paths rather than using the dynamic routing algorithms which need more computation time for each node.

1. Mathematical Model

Input: G is a simple directed graph, M specifies set of node for the case of the weight of an edge increased/ decreased. Original weight wt and changed weight wt'.

Output: The updated SPT rooted at s in the updated graph G.

System:

Let, Set of nodes S = {l1, l2,…., lk} V

Where,

nodes lk and V are is set of nodes.

Let G = (v, E, wt) denote a directed graph constructed using set of nodes.

Where,

V is the set of nodes,

E is the set of edges

wt represents the weight of each edge in E in the graph. Given an edge e : i → j

Where,

i is the source node and j is the destination node of e.

Wt' (edg) is used to show the new weight of edge e.

a temporary SPT with S (G) as the root is maintained. When the update process is terminated, the temporary tree becomes the final new SPT.

Let d = D(i) + wt'(edg) - D(j)

Where,

d represents the increment value to node j if the shortest path to node j through the edge e. Sometimes d can be negative if the weight of edge e becomes smaller.

2. Dynamic SPT:

Dynamic SPT algorithm is as below:

Input: G is a simple directed graph, M specifies set of node for the case of the weight of an edge increased/ decreased. Original weight wt and changed weight wt0.

Output: The updated SPT rooted at s in the updated graph G.

Step 1: From G = (V, E) having SPT constructed.

Step 2: wait until one edge edg : i → j changed its weight from wt(edg) to wt'(edg). Apply Dynamic routing algorithm and update routing table.

Step 3: Dynamic routing

Find shortest paths by dynamic routing

G=(V,E) SPT algorithm of only changed portion of SPT

Initialization des(e) are updated

following the sequence of DFS from node e in SPT

// all descendants of updated

Remove edges from Q which have end nodes belonging to des(e)

Update the old information in Q

Obtain a Temporary SPT

While(des(e))

{des(e), mis_inc} ← extract(M)

If v has incoming links between des(e), then

if D(i) from incoming link > D(j) from inner nodes,

then D(i) = D(j)

endif

endif

Update the routing .

Alternatively, dynamic routing algorithms should be applied to compute the shortest paths when some links have new weights near the end node. The reason that dynamic routing algorithms are applied in this case is that there are only a few nodes which have to be computed near the end node. In this case, using dynamic routing algorithms to re-compute only the nodes affected by old shortest paths is better than using the static routing algorithms which re-compute every node

PERFORMANCE EVALUATION

The performance of the Dynamic SPT algorithm is compared to classical SPT algorithm. The number of nodes, the changed link weights were used for the input parameters in the simulations. The computation time calculated with the new algorithm and old static algorithm is compared for the case of one edge weight change. New algorithm introduces the node list M when the weight of an edge increased and directly updates the node set des(j) when the weight of an decreased. These methods greatly reduce the time to enqueue and dequeue edges from Q and consequently have less time to search for the edge with the minimum value in Q. In the simulation (a program which calculates the computation time when weight changes using algorithm) for a specific network size, 100 continuous weight changes is tested based on one generated graph. The performance comparison between the new algorithm and old SPT for different ranges of edge weight is shown in figure 4 and 5.

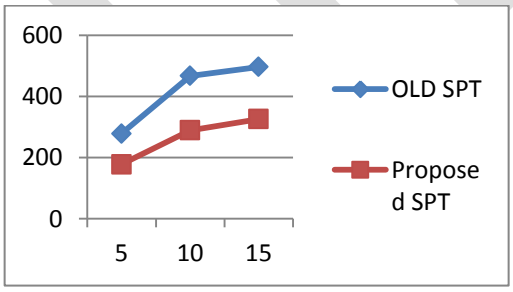

Figure.4. Computation Time over proposed SPT vs old SPT (time taken vs no of nodes weight changes)

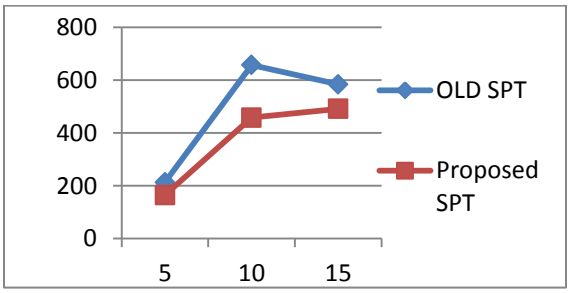

Figure.5. Search Time in network over proposed SPT vs old SPT

CONCLUSION

Proposed Dynamic algorithm present an efficient shortest paths construction used to minimize the total execution time. Less total execution time provide to reduction in packet loss. As shown in the comparison output, the proposed algorithm provides a better performance when it compared to the older method in terms of the computation time of the shortest path tree. This algorithm minimize the calculation time and only smaller number of changes are made to SPT structure. Thus, it removes the disadvantage caused by static algorithms for SPT update like updating of the whole SPT if there is any change.

ACKNOWLEDGMENT

I would like to articulate deep gratitude to author Prof. Soumitra Das, Head of Computer Engineering Department who has always been a source of motivation and firm support for carrying out the paper. I would also like to convey our sincerest gratitude and indebtedness to all other faculty members and staff of Department of Computer Engineering, D Y Patil School of Engineering, Pune, who bestowed their great effort and guidance at appropriate times without which it would have been very difficult on our Paper Work.

REFERENCES:

[1] J. Moy, OSPF version 2. Internet Draft, RFC 2178, 1997.
[2] B. Fortz and M. Thorup, Optimizing OSPF/IS-IS weights in a changing world, IEEE Journal on Selected Areas in Communications, vol. 20, pp. 756– 767, May 2002.
[3] E. Dijkstra, A note two problems in connection with graphs, Numerical Math., vol. 1, pp. 269–271, 1959.
[4] X. Xiao and L. Ni, Reducing routing table computationcost in OSPF, in Proc. Internet Workshop, IWS'99, pp. 119–125, 1999.
[5] M. Noto and H. Sato, A method for the shortest path search by extended Dijkstra algorithm, 2000 IEEE international Conference on Systems, Man, and Cybernetics, vol. 3, pp. 2316–2320, 2000.
[6] V. King, Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs, in IEEE Symposium on Foundations of Computer Science, pp. 81–91, 1999.
[7] E. Nardelli, G. Proietti, and P. Widmayer, Swapping a failing edge of a single source shortest paths tree is good and fast, Algorithmica, vol. 35, pp. 56–74,2003.
[8] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, Fully dynamic shortest paths in digraphs with arbitrary arc weights, Tech. Rep. ALCOMFT-TR-01-31 of the EC-IST Project ALCOM-FT, March 2001.
[9] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, Fully dynamic output bounded single source shortest path problem, in Proc. 7th Annu. ACM-SIAM Symp. Discrete Algorithms, (Atlanta, GA), pp. 212– 221, 1998.
[10] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, New dynamic algorithms for shortest path tree computation, IEEE/ACM Trans. Networking, vol. 8, pp. 734–746, Dec. 2000.
[11] T. H. Cho, J. W. Kim, B. J. Kim, W. O. Yoon and S. B. Choi, "A Study on Shortest Path Decision Algorithm for Improving the Reliability of Dynamic Routing Algorithm", Journal of the Korean Institute of Information Scientists and Engineers, vol. 38, (2011), pp. 450-459.
[12] V. Eramo, M. Listanti and A. Cianfrani, "Design and Evaluation of a New Multi-Path Incremental Routing Algorithm on Software Routers", IEEE Transactions on Network and Service Management, vol. 5, (2008), pp.188-203.
[13] S. K. Lee, J. W. Jang, S. J. Jang and J. Y. Shin, "Development and Performance Analysis of ABR-DBA Algorithm for Improve Network Performance", International Journal of Future Generation Communication and Networking, vol. 1, (2008), pp. 1-6