International Journal of Research

International Journal of Research (IJR)

e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 10, October 2015

Available at http://internationaljournalofresearch.org

A Novel Approach for Resilient Multipath Routing by Using Independent Directed Acyclic Graphs

Student: **Aakunuru.Rahul**

Guide: **Kalyan**

(Vellore Institute of Technology, Vellore**)**

Abstract—

In this paper, we introduce the concept of Independent Directed Acyclic Graphs (IDAGs) to achieve resilient multipath routing. The algorithm developed in this paper: (1) achieves multipath routing; (2) guarantees recovery from single link failure. The multipath routing scheme consists of an algorithm to determine a set of multiple disjoint or partially disjoint paths and a mechanism for distributing traffic over a multipath route to reduce the traffic load on a congested link. Multipath routing is capable of aggregating the resources of multiple paths and reducing the blocking capabilities in QoS oriented networks, allowing data transfer at higher rate when compared to single path. To improve the failure resiliency without jeopardizing the routing stability, we propose a local rerouting based approach called failure insensitive routing. Under this approach, upon a link failure, adjacent router suppresses global updating and instead initiates local rerouting. We demonstrate that the proposed approach provides higher service availability than the existing routing schemes.

I.INTRODUCTION

Multipath routing is used to find out multiple pairs of paths between source and destination. The scheme provides robustness [2], load balancing [3], and security [6]. Techniques developed for multipath routing based on employing multiple spanning trees or directed acyclic graphs (DAGs) [7]. The packet needs to be dropped when adjacent node are not available. This dropping takes place because of potential looping of packets when transferred from one routing table to another. It Provides fast recovery from single-link failures provide more than one forwarding edge to route a packet to a destination. The techniques categorized on the basis of the nature in which the backup edges are employed. In [8], the developed a method to augment any given tree rooted at a destination with “backup forwarding ports.” The packet is received from the node attached to the default forwarding edge for the destination; the packets are rerouted on the backup ports. In [9], the framework for IP fast reroute detailing three candidate solutions for IP fast reroute that have all gained considerable attention. These are multiple routing conFigureurations (MRCs) [10], failure insensitive routing (FIR) [11], [12], and tunnelling using Not-via addresses [13]. The common feature of all these approaches is that they employ multiple routing tables. However, they differ in the mechanisms employed to identify which routing table to use for an incoming packet. The readers are referred to [14] for a detailed description of the above techniques. It is certainly possible to use fast recovery techniques (irrespective of whether they guarantee recovery from single link failure or not) for multipath routing. However, all the above techniques require a significantly large number of routing tables, hence a large number of additional bits in the packet header. One approach that offers resiliency to single-link failure and provides multipath routing to some degree is "colored trees" [15], [16]. In this approach, three trees are constructed per destination node such that the paths from any node to the root on the two trees are disjoint. The trees may be constructed to obtain link-disjoint or nodded is joint paths if the network is two-edge or two-vertex connected, respectively. This approach is similar to those employing multiple routing tables, except that only two tables are required. Every packet may carry an extra bit in its header to indicate the tree to be used for routing. This overhead bit may be avoided by employing a routing based on the destination address and the incoming edge over which the packet was received, as every incoming edge will be present on exactly one of the trees. The colored tree approach allows every node to split its traffic between the three trees, thus offering disjoint multipath routing by using red tree approach. In addition, when a forwarding link on a tree fails, the packet may be switched to the other tree by using blue tree approach; else it uses a green tree approach which is used to recover from only a dual-link failure. A packet may be transferred from one tree to another at most once as the coloured tree approach. Here the colour of the tree to be used for routing may be obtained from the incoming interface over which the packet was received. Thus, every packet needs to carry only one bit overhead.

II. Independent Directed Acyclic Graphs

We consider a network with a set of nodes and links denoted by N and L, respectively. We assume that links are bidirectional in nature, which may be realized using two unidirectional links. We denote a bidirectional link between nodes i and j as i—j, while the directed link from i to j is denoted by i→j . When a link fails, we assume that both directed edges i→j and j→I have failed. A. Resilient Routing With IDAGs The network is assumed to employ link-state protocol, hence every node has the view of the entire network topology. Every node computes two DAGs, namely red and blue, for each destination and maintains one or more forwarding entries per destination per DAG. The DAGs may be used in two different ways to achieve resilient routing. In the first approach, referred to as Red DAG first (RDF), the packets are assumed to be forwarded on the red DAG first. When no forwarding edges are available on the red DAG, the packet is transferred to the blue DAG. When no blue forwarding edges are available, the packet is dropped. The DAG to be employed for routing is carried in an overhead bit (DAG bit) in every packet header. In the second approach, referred to as Any DAG first (ADF), a packet may be transmitted by the source on the red or blue DAG. In addition to the DAG bit, every packet also carries an additional bit that indicates whether the packet has been transferred from one DAG to another (Transfer bit). A packet is routed on the DAG indicated in its packet header. If no forwarding edges are available in that DAG and if the packet has not encountered a DAG transfer already, it is transferred to the other DAG. If no forwarding edges are available on the DAG indicated in the packet header and the packet has already encountered a DAG transfer, the packet is dropped. In both of the approaches described above, a node may forward the packet along any of the available forwarding edges in the DAG indicated in the packet header

III. Constructing Node-Independent DAGS

Two-vertex-connectivity is the necessary and sufficient requirement for constructing two node-independent DAGs utilizing all the edges except those emanating from the given destination node. This necessary part of the requirement follows directly from the condition required for constructing two node-independent trees—a special case of DAG. We show the sufficiency part of the requirement by constructing the desired DAGs. A. Procedure for NI-DAGs Construction

1. Initialize R and B to contain only the root node d. Initialize the partial order of the nodes on the red and blue DAGs to the empty set.

2. Find a cycle (d,v1,… vk ,d). Let vk → vk-1→ … →v1→d be the red chain. Add the blue chain to B and the red chain to R. update the precedence relation with d<v1<v2< … <v1<v2< … the red DAG.

Find a path (x,v1,….,vk ,y) that connects any two distinct nodes x and y on R and any k nodes not on R,k ≥1, such that x<v1<v2< … →vk →y be the blue chain. Add the blue chain to B and the red chain to R. Update the precedence relation with x<v1<v2< … <v_k<y on the red DAG. Note that: (i) if y=d, then vk<y is ignored; and (ii) if y=d and/or x=d, y→vk and/or x→v1 in the red and blue chains above, respectively

4. If B does not span all the nodes in G, goto step 3.

5. Compute a global order that is consistent with the partial on the red DAG. We denote x precedes y in the global order as x<y.

6. For every link i-j (i≠d,j≠d) that is not on the DAGs: • If i<j then add i←j to the red DAG and i→j to the blue DAG. Otherwise, add i←j to the blue DAG and i→j to the red DAG. • For every edge i→d that is not on the DAGs, add i→d either to the red or the blue DAG randomly. The above procedure is given to construct multiple link independent DAGS. These DAGs provide effective multipath routing.

IV. Constructing Link-Independent DAGS

Two-edge connectivity is a necessary and sufficient condition for constructing two link-independent DAGs. Similar to the requirement of node-independent DAGs, the necessary part of the requirement follows from the independent tree construction.

A. Procedure for LI-DAGs Construction

1. Divide the network into two vertex connected (2V) components

2. In each 2V component, select the unique articulation node through which every path from any node in that component must traverse to reach d. We refer to this articulation node as the root node of the component. In the component that contains node d , we assume that the root node of the component is node d itself

3. In each 2V component, construct two node independent DAG to the root node of that component.

4. Merge all node independent DAG to obtain the Link independent DAGs The above procedure is given to construct multiple link independent DAGS. These DAGs provide effective multipath routing [1].

V. DESIGN PROCESS

A. System Architecture Diagram

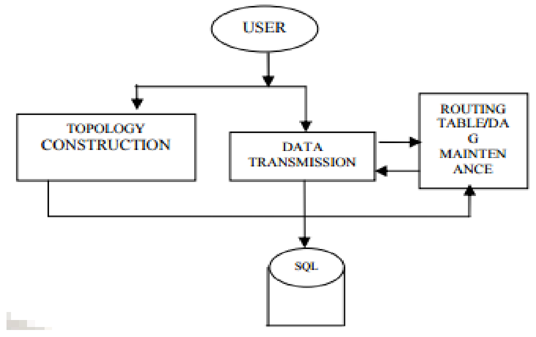

Fig 1. System Architecture Diagram

B. Modules Description

1. Topology Construction

This module is used to construct the topology. The user gives the number of node used to construct the topology. The node is added in given name, IP address and port number of that node. Unique nodes are created so that it can be logged in separately. After adding the node, the source node name, neighbor node name and right of that path will be given for path connection. The node details are stored in database table called Node details. Routing details are stored in the routing table.

2. Multipath Routing

The network is assumed to employ link-state protocol; hence every node has the view of the entire network topology. Every node computes DAGs, for each destination and maintains one or more forwarding entries per destination per DAG. DAG to be employed for routing is carried in an overhead bit (DAG bit) in every packet header. Any DAG first (ADF), a packet may be transmitted by the source DAG. In addition to the DAG bit, every packet also carries an additional bit that indicates whether the packet has been transferred from one DAG to another (Transfer bit). A packet is routed on the DAG indicated in its packet header. If no forwarding edges are available in that DAG and if the packet has not encountered a DAG transfer already, it is transferred to the other DAG.

3. Node Independent DAG

Two-vertex-connectivity is the necessary and sufficient requirement for constructing two node in dependent DAGs utilizing all the edges except those emanating from the given destination node. This necessary part of the requirement follows directly from the condition required for constructing two node-independent trees – a special case of DAG. Initialize the partial order for the nodes on the two DAGs. Compute the first cycle to be augmented. Compute successive paths to be augmented. The path starts and ends at distinct nodes that are already added to the DAGs; hence the paths from every node to the root of the DAG are node-disjoint. Note that the difference between the path augmentation employed for DAG construction here and that employed for tree construction.

4. Link Independent DAG

Two-edge connectivity is a necessary and sufficient condition for constructing two link-independent DAGs. Similar to the requirement of node-independent DAGs, the necessary part of the requirement follows from the independent tree construction. The procedure to construct two link independent DAGs. Divide the network into two vertex- connected (2V) components. A node may appear in more than 2Vcomponent and the removal of such a node (articulation node) would disconnect the graph. In addition, any two 2V-components may share at most one node in common. Given a destination node d, identify the root node for every component the unique node through which every path connecting a node in that component and d must traverse. In components that contain node d, the root node is assumed to be d.

VI. CONCLUSION

In this paper, we introduced the concept of independent directed acyclic graphs (IDAGs) and developed a methodology for resilient multipath routing using two IDAGs. We developed polynomial time algorithms to construct node-independent and link-independent DAGs using all possible edges in the network. The IDAGs approach was evaluated on four real-life network topologies and compared with ITrees and multiple pairs of colored (independent) trees approaches to prove the Validity of the algorithm. Through simulations, we showed that the IDAGs approach performs significantly better than the independent trees approach in terms of increasing number of paths offered, reducing the probability of a two-link failure disconnecting a node from the destination, and average link load. Even, a simulation result showed that the trees based on the shortest paths on the IDAGs have better performance than that of the ITrees approach since the average shortest path length on the IDAGs is shorter than the average path length on the ITrees. Multiple pairs of colored trees approach is better in terms of the product of the number of critical links and average link load compared to the ITrees and IDAGs

VII. REFERENCES

[1] S. Cho, T. Elhourani, and S. Ramasubramanian, “Resilient multipath routing with independent directed acyclic graphs,” in Proc. IEEE TRANSACTION ON NETWORKING ,vol. 20, no. 1, february 2012, pp. 153-162.

[2] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, “A framework for reliable routing in mobile ad hoc networks,” in Proc. IEEE INFOCOM, Apr. 2003, pp. 270–280.

[3] P. P. Pham and S. Perreau, “Performance analysis of reactive shortest path and multi-path routing mechanism with load balance,” in Proc. IEEE INFOCOM, 2003, pp. 251–259.

[4] J. Tsai and T. Moors, “A review of multipath routing protocols: From wireless ad hoc to mesh networks,” in Proc. ACoRN Early Career Res. Workshop Wireless Multihop Netw., Jul. 17–18, 2006, pp. 17–22.

[5] S. Murthy and J. Garcia-Luna-Aceves, “Congestion-oriented shortest multipath routing,” in Proc. IEEE INFOCOM, Mar. 1996, vol. 3, pp. 1028–1036.

[6] W. Lou, W. Liu, and Y. Fang, “A simulation study of security performance using multipath routing in ad hoc networks,” in Proc. IEEE Veh. Technol. Conf., Oct. 2003, vol. 3, pp. 2142–2146.

[7] G. Lee and J. Choi, “A survey of multipath routing for traffic engineering,” 2002 [Online]. Available: http://academic.research microsoft.com/Publication/10842993/a-survey-of-multipath-routingfor-traffic-engineering

[8] K. Xi and J. Chao, “IP fast rerouting for single-link/node failure recovery,” in Proc. BROADNETS, Internet Technol. Symp., Sep. 2007, pp. 142–151.

[9] M. Shand and S. Bryant, “IP fast reroute framework,” IETF Internet Draft draft-ietf-rtgwg-ipfrrframework-08.txt, Feb. 2008.

[10] A. Kvalbein, A. F. Hansen, T. Ĉiĉić, S. Gjessing, and O. Lysne, “Fast IP network recovery using multiple routing conFigureurations,” in Proc. IEEE INFOCOM, Apr. 2006, pp. 1–11.

[11] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, “Proactive vs. reactive approaches to failure resilient routing,” in Proc. IEEE INFOCOM, Mar. 2004, vol. 1, pp. 176–186.

[12] S. Nelakuditi, S. Lee, Y. Yu, and Z.-L. Zhang, “Failure insensitive routing for ensuring service availability,” in Proc. IWQoS, Jun. 2003, pp. 287–304.

[13] S. Bryant, M. Shand, and S. Previdi, “IP fast reroute using not-via addresses,” Internet Draft draftietf-rtgwg-ipfrr-notvia-addresses-02.txt, Feb. 2008.

[14] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. Hansen, “Fast recovery from dual-link failures in IP networks,” in Proc. IEEE INFOCOM, Rio de Janeiro, Brazil, Apr. 2009, pp. 1368–1376.

[15] S. Ramasubramanian, M. Harkara, and M. Krunz, “Distributed linear time construction of colored trees for disjoint multlipath routing,” in Proc. IFIP Netw., May 2006, pp. 1026–1038.

[16] G. Jayavelu, S. Ramasubramanian, and O. Younis, “Maintaining colored trees for disjoint multipath routing under node failures,” IEEE/ACM Trans. Netw., vol. 17, no. 1, pp. 346–359, Feb. 2008.