International Journal of Mathematics and Soft Computing
Vol.3, No.3. (2013), 1 - 5.

ISSN Print : 2249 – 3328
ISSN Online: 2319 – 5215

Weighted Independent Directed Acyclic Graphs for resilient and efficient multipath routing

M. Ramakrishnan
Professor and Head,
Department of Information Technology,
Velammal Engineering College, Chennai, INDIA.

Sowmya Swaminathan
M.Tech., Information Technology,
Velammal Engineering College, Chennai, INDIA.
E-mail: sowmyaforever@gmail.com

Jayaganesan Chandrasekaran
Project Leader, Triad Software Private Ltd,
Chennai, INDIA.

Abstract

In order to achieve resilient multipath routing with efficient path choosing dynamically, we introduce the concept of Weighted Independent Directed Acyclic Graphs (WIDAGs). In a given network we develop polynomial-time algorithms to compute link-independent and node-independent DAGs. We use Heuristic methods to dynamically choose the maximum profitable (least cost) route upon a single link failure. The algorithm developed in this paper takes the advantage of the resiliency provided by IDAG and in addition, improves efficiency by choosing least cost path dynamically.

Keywords: Independent Directed Acyclic Graphs (IDAGs), failure recovery, weighted independent graphs, multipath routing, Dijkstra's algorithm, shortest path routing.

AMS Subject Classification (2010): 05C20, 05C85.

1 Introduction

Internet provides platform for a variety of applications and all these applications have the need for robustness and bandwidth. Various techniques have been developed for achieving better robustness of the network. Multipath routing is a technique in which the packets are routed along multiple paths between the source and the destination. This scheme promises security and load balancing. Recovery techniques based on multipath routing use more than one route to the destination and employ multiple spanning trees or DAGs. When there is a failure in one of the routes, the packets are routed via the alternate routes. In [6], every node has backup forwarding edges. When the default forwarding edge fails, alternate edge is chosen from the backup forwarding edges. In [5], the authors present a framework for IP fast reroute detailing three candidate solutions for IP fast reroute that have all gained considerable attention. These are multiple routing configurations (MRCs), failure insensitive routing (FIR) and tunneling using Not-via addresses (Not-via). The common feature of all these approaches is

that they employ multiple routing tables. Colored trees approach for multipath routing is explored in [3], [2]. In this approach, two colored trees are constructed that are link or node disjoint with each other. This approach is similar to employing multiple routing tables except that only two tables are required in this case. The Independent DAGs, introduced in [4], are similar to the colored trees. The trees can utilize a maximum of 2(|N|-1) directed edges for routing from a source to a destination, where |N| is the number of nodes in the network. The IDAGs overcome this inherent limitation of the trees by utilizing all possible edges available in the network except for those emanating from the destination and those limited by the underlying network topology. [4].

The IDAG routing guarantees single link failure recovery. What is not considered is the efficiency of the alternate route chosen. There are many Quality of Service (QoS) based techniques available to choose the least cost path among the available routes. We consider here the Dijkstra's algorithm [1] to perform the least cost path computation. We introduce the concept Weighted Independent Directed Acyclic Graphs (WIDAGs), to include the link cost pertaining to the network characteristics to the IDAGs.

In this paper, we develop an efficient routing mechanism over the resilient routing technique IDAG. We introduce the concept of WIDAGs, an extension of IDAGs. The network characteristics of delay, bandwidth, processing power and the like are consolidated and represented as a cost for each link of the network. IDAGs are constructed using linear time algorithms and failure routing is appended with efficient least cost path computation algorithms. The algorithms for construction of IDAGs are proven to utilize maximum possible edges of the network. The Dijkstra's algorithm chooses the most efficient path heuristically.

2 Weighted Independent Directed Acyclic Graph

Consider a network with a set of nodes and links denoted by N and L, respectively. Assume that the links are bidirectional in nature, which may be realized by using two unidirectional links. A bidirectional link between nodes i and j is denoted by i – j, while the directed link from i to j is denoted by i→j. When a link fails, it is assumed that both directed edges i→j and j→i have failed. DAG is rooted at d if d is the only node in the DAG that has no outgoing edges. Every other node has at least one outgoing edge. If the sequence of edges starting from any node is traversed, the path will terminate at node d and will be loop-free.

Consider two directed acyclic graphs that are rooted at d. The two DAGs are said to be link-independent if for every node s Є N, s ≠d, any path from s to d on one DAG is link disjoint with any path from s to d on the other DAG. Similarly, the two DAGs are said to be node-independent if for every node s Є N, s ≠d, any path from s to d on one DAG is node-disjoint with any path from s to d on the other DAG.

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.

Consider a network G(N;L) composed of a set of nodes N and a set of links L. The links are assumed to be bi-directional. An arc i → j represents a directed link between nodes i to j. Let W_{ij} denote the cost of link i–j, such as delay on a link. Given a destination node $d \in N$, we construct and maintain two trees R and B (referred to as the red and blue trees, respectively) that are node (link) independent and rooted at d such that the average path length from a node to the drain is minimized.

A network must be two-node-connected (two-edge connected). In addition, the network must remain two-connected to reconstruct the colored trees after a failure. If a solution to the link-disjoint (node disjoint) problem is required after k arbitrary node (link) failures, then the network must be k + 2 node- (edge-) connected. While there exists a set of m nodes (links), where m > k, whose removal would result in a less than 2- node (edge) connected network, not all combinations of m node (link) removal will result in a lack of two-connectivity. Figure 1 shows the transition from IDAG to WIDAG for an example network.

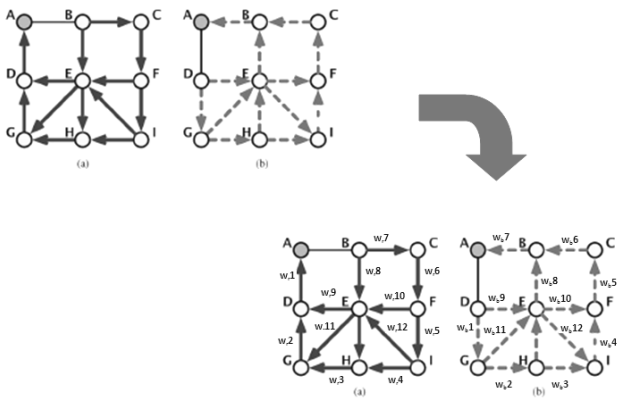

Figure 1: DAG *to* WIDAG *for an example network*.

The efficient re-routing requires the link properties like delay, traffic, bandwidth and the like to be considered. These parameters are aggregated and each link in the network is given a weight. Thus the IDAGs formed from such networks are Weighted Independent Directed Acyclic Graphs. Thus, the Weighted Independent Directed Acyclic Graphs (WIDAGs) are along with weights for the links of the corresponding independent directed acyclic graph.

Also, when there is a link i-j with weight w_{ij} in the network, both the directed links i ← j and i → j are assigned the same weight w_{ij}, for simplicity. This is also because most of the static parameters will not vary between both directions of the link of the original network.

3 Construction of WIDAG

We construct the IDAGs using the procedure discussed in [4]. Accordingly, two-vertex-connectivity is the necessity and the sufficient condition for construction of two node-independent IDAGs. We first compute the base independent trees by using path augmentation techniques. We then add the edges that are not present in either of the DAGs and maintain precedence relation among the nodes in both the trees. From [4], A node x precedes y, denoted by $x \prec y$, on a DAG if node y uses node x in at least one of its paths to d. The partial order on a DAG may be viewed simply as

reachability on the DAG, that is, $x \prec y$ implies that x is reachable on the DAG by y. This relationship is the key to the construction as it avoids any cycle formation, hence the DAGs. The procedure for constructing WIDAG is given in Figure 2. We extend this procedure with the least path computation algorithm. This way, on any node failure, the search for an alternative path will be combined with Dijkstra's algorithm to make the most optimal choice. The Dijkstra's algorithm is given below:

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm assigns some initial distance values and will try to improve them step by step.

1. Assign a tentative distance value to every node: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all the nodes, unvisited. Set the initial node as current. Create a set of the unvisited nodes called the unvisited set consisting of all the nodes except the initial node.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. For example, if the current node A is marked with a tentative distance of 6, and the edge connecting it with a neighbour B has length 2, then the distance to B (through A) will be 6+2=8. If this distance is less than the previously recorded tentative distance of B, then overwrite that distance. Even though a neighbour has been examined, it is not marked as "visited" at this time, and it remains in the unvisited set.
4. When we consider all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again; its distance recorded now is final and minimal.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal), then stop. The algorithm has finished.
6. Set the unvisited node marked with the smallest tentative distance as the next "current node" and go back to step 3.

The computational complexity for constructing the node-independent IDAG is O(|L|), where L is the number links in the base network [4]. The Dijkstra's algorithm has a computational complexity of O(|L|+|N|log|N|).

Algorithm to construct WIDAG

Consider a network of **N** nodes and **E** links.

1. Get the network layout in the form of adjacency matrix.
2. The adjacency matrix has entries as
   { **w** when edge present
   0 when edge absent } where **w** is the weight of the edge.

3. Check if the entered network is two-vertex connected.
4. Get the source node **s** and the destination node **d** in the network.
5. For the given source and destination node, we construct a DAG **R** from the underlying undirected graph, utilizing all its edges.
6. We construct the corresponding IDAG **B** of the DAG **R** using N-IDAG construction method in [4] .
7. For a given failure node in **R**, we find an alternate shortest route from the source to the destination in **R**.
8. We switch to **B** in case an alternate route is unavailable in **R**.

Figure 2: WIDAG - *construction algorithm.*

4 Conclusion

The previous studies on resilient multipath routing has been done by several authors mainly involving colored trees and Independent Directed Acyclic Graphs (IDAGs) . An interesting idea arises when the networks under consideration are weighted. In such situations, we have to search for a minimum weighted alternate route in the case of single link failure. Our attempt is to amalgamate the available ideas with the shortest path algorithm to find a solution to this problem. The system can be further studied on suitability for a multi-link failure recovery routing.

References

[1] E. Dijkstra, *A note on two problems in connection with graphs*, Numerical Mathematics, Vol. 1, (1959), 269-271.

[2] 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, (2008), 346-359.

[3] S. Ramasubramanian, M. Harkara and M. Krunz, *Distributed linear time construction of colored trees for disjoint multipath routing*, In Proc. IFIP Netw, (2006), 1026-1038.

[4] Sangman Cho, Theodore Elhourani, and SrinivasanRamasubramanian, *Independent Directed Acyclic Graphs for Resilient Multipath Routing*, In IEEE/ACM transactions on Networking, Vol. 20, No. 1(2012).

[5] M. Shand and S. Bryant, *IP fast reroute framework*, IETF Internet Draft draft-ietf-rtgwg-ipfrr-framework-08.txt, (2008).

[6] K. Xi and J. Chao, *IP fast rerouting for single link/node failure recovery*, In Proc. BROADNETS, Internet Technol. Symp.,(2007), 142-151.