

Let More Nodes Have a Second Choice

Haijun Geng^{*‡}, Xingang Shi^{†‡}, Xia Yin^{*‡}, Zhiliang Wang^{†‡}, Han Zhang^{*‡}, Jiangyuan Yao^{*‡}

^{*}Department of Computer Science & Technology, Tsinghua University

[†]Institute for Network Sciences and Cyberspace, Tsinghua University

[‡]Tsinghua National Laboratory for Information Science and Technology (TNList)

Abstract—Current intra-domain routing protocols computes only shortest paths for any pair of nodes, which cannot provide good fast reroute when network failures occur.

Multipath routing can be fundamentally more efficient than the currently used single path routing protocols. It can significantly reduce congestion in network by shifting traffic to unused network resources. This improves network utilization and provides load balancing.

To enhance failure resiliency, we propose a new scheme, More Nodes Have At Least Two Choices (MNTC), where the goal is how to maximize the number of nodes that have at least two next-hops towards their destinations.

We evaluate the algorithm in a wide space of relevant topologies, and the results show that it can achieve good reliability while keeping low stretch.

Keywords—multipath routing; reliability; stretch

I. INTRODUCTION

Current deployed intra-domain link-state routing protocols, such as OSPF and ISIS, are concerned with computing shortest paths towards a destination. Therefore, they cannot provide good connectivity under frequent network failures. Multipath routing [1] has been proposed to deal with the problem. It not only improves the resilience to route failures, but also provides other benefits such as better load balancing, enhanced bandwidth and improved security. To improve the reliability and load-balancing capabilities of a network, it is important that there is more than one next-hop available for as many source-destination pairs as possible, while keeping path stretch as small as possible. We evaluate the performance of our algorithm using synthetic topologies. The evaluation results demonstrate that it can provide high reliability.

The rest of paper is organized as follows. Section II introduces our multipath routing algorithm in detail. Section III evaluates its performance in a variety of network topologies and finally Section IV concludes the paper.

II. COMPUTING MULTIPLE NEXT HOPS

A. Problem Formulation

A network is modeled as a undirected connected graph $G=(V, E)$, where V and E are respectively the set of nodes and edges in the network. For a specific destination d , T_d be a shortest path tree (SPT) rooted at d , let $R_d(V, E)$ be a directed acyclic graph (DAG) rooted at d , where all links are included.

R_d ¹ defines a destination-based routing. Let $N(s, d)$ be the next-hop set from node s to node d .

Given a network graph G , our objective is how to compute R_d so that it can maximize the number of nodes that have at least two next-hops towards destination d . Since the resulting routing is compatible with normal shortest path routing, T_d is included in the R_d .

Now we define the MNTC problem formally as follows:

$$\text{Maximize} \quad \sum_{s,d \in V} m(s, d) \quad (1)$$

where

$$m(s, d) = \begin{cases} 1 & |N(s, d)| \geq 2 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$\text{s.t.} \quad T_d \subseteq R_d \quad (3)$$

Definition 1. With respect to a R_d , a relation ($<_{R_d}$) on the nodes is defined such that $x <_{R_d} y$ if and only if node x is added to R_d prior to node y .

Lemma 1. If all nodes agree on a R_d rooted at the destination d , any node c can choose its neighbor x as its next hop towards d without introducing any loop, as long as $x <_{R_d} c$.

This lemma can be simply proved by the fact that, for any forwarding path, i.e., $\langle u_1, u_2, \dots, u_k \rangle$, we have $u_k <_{R_d} u_{k-1} <_{R_d} \dots <_{R_d} u_1$. So there exists a strict partial order between any two nodes on the path, and no node can appear twice on the path, which means there is no loop.

Algorithm 1: MNTC

Input: Network $G(V, E)$

Input: Destination d

```

1  $d.seq \leftarrow 1$ ;
2  $U = \{d\}$ ;
3 Compute  $T_d$ ;
4  $i \leftarrow 2$ ;
5 while  $i \leq |V|$  do
6    $M = \{m \in V \setminus U \mid \forall (m, n) \in T_d, n \in U\}$ ;
7   Select a node  $u \in M$ ;
8    $u.seq \leftarrow i$ ;
9    $U = U \cup \{u\}$ ;
10   $i \leftarrow i + 1$ ;
```

¹We use R_d as a simplification of $R_d(V, E)$.

B. Algorithm

An algorithm for computing multiple next-hops towards destination d is given in Algorithm 1. We assign a sequence number attribute to each node denoted by seq . Let U denote the set of nodes which have been already added to the R_d , M denote the set of nodes such that within the T_d , which have direct links with the U , and $Deg(u, U)$ be the number of edges between a node u and a set of nodes U . At first, the node d is added to the R_d and included in the set U (lines 1-2). We further build a shortest path tree rooted at d using Dijkstra's algorithm, and assign a sequence number to each node to indicate the order in which these nodes are added to the tree (line 3). When the construction of T_d is finished, all the nodes have a unique sequence number, which are also stored in their respective seq .

Then, we compute a R_d according to the tree constructed above. To construct the R_d , we go through several iterations. In each iteration, we examine the nodes in the M and select one to add to the R_d . To determine which node will be added to the R_d , for each node u in the M , we compute the number of edges between a node u and a set of nodes M . In order to let more nodes have at least two next-hops to destination d while keeping path stretch as small as possible. The node u is chosen as current node added to the R_d (lines 6-7) according to:

Definition 2. (The Node Selection Rule)

- 1) Select the one u if $Deg(u, M) \geq 2$; or
 - 1.1) if multiple such u exist, select the one with the smallest sequence number
 - 1.2) if such u does not exist, select the one with the smallest sequence number

At last, the sequence number and the set U will be updated when the node u is added to the R_d (lines 8-9).

III. PERFORMANCE EVALUATION

In this section, we present our evaluation methods as well as results on synthetic topologies. We compare the results achieved by MNTC against Downstream Condition (DC) [2] and Loop Free Alternates (LFA) [2]. We use BRITE to generate a variety of topologies.

A. Reliability

We use coverage as our main evaluation criterium. Coverage can be defined as:

$$coverage = \frac{\sum_{s,d \in V} m(s,d)}{|V| * (|V| - 1)} \quad (4)$$

Fig. 1 shows the coverage achieved by each algorithm on different topology sizes. From Fig. 1, we can come to the conclusion that as the topology size increases, coverage increases too. Our algorithm has a clear advantage over other three algorithms in all cases.

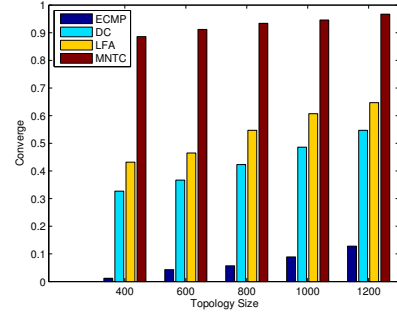


Fig. 1: Average Node Degree=4

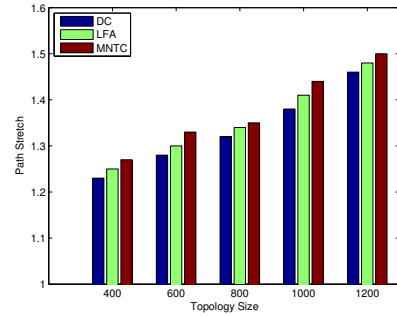


Fig. 2: Average Node Degree=4

B. Path Stretch

Path stretch is defined as the ratio of the cost on all the paths to the ratio of the shortest paths between all source-destination pairs. As shown in the Fig. 2, our algorithm is still comparable to those of DC and LFA.

IV. CONCLUSION

In this paper, we propose a DAG-based multipath routing algorithm. The goal of our algorithm is to maximize the reliability in a network with traditional hop-by-hop forwarding. Our algorithm can achieve good performance in increasing the network coverage while keeping low stretch, as demonstrated by our evaluation results. We believe networks can be made more efficient and reliable by adopting this mechanism.

V. ACKNOWLEDGE

This work is partially supported by the National Natural Science Foundation of China (Grant No. 61202357).

REFERENCES

- [1] Y. Ohara, S. Imahori, and R. Van Meter, "Mara: Maximum alternative routing algorithm," in *Proceedings - IEEE INFOCOM*, 2009, pp. 298 – 306.
- [2] E. A. Atlas and E. A. Zinin, "Basic specification for ip fast reroute: Loop-free alternates, rfc 5286," 2008.