Real Time Control Plane Verification

Yifan Li
Research Institute of Information Technology (RIIT)
Beijing, China
liyifan18@mails.tsinghua.edu.cn

Jake Jia
Research Institute of Information Technology (RIIT)
Beijing, China
jcj18@mails.tsinghua.edu.cn

Xiaohe Hu
Research Institute of Information Technology (RIIT)
Beijing, China

Jun Li
Research Institute of Information Technology (RIIT)
Beijing, China

ABSTRACT

In cloud datacenters, network configuration changes frequently, thus fast control plane verification is demanded to validate the changes before deployed. However, existing control plane verification tools are all based on static analysis of network configuration, so that the configuration need to be reverified completely when edited. Inspired by the real time data plane verification, we propose an incremental control plane verification. Its implementation based on an existing work demonstrated supeior performance.

CCS CONCEPTS

• **Networks → Network management**.

KEYWORDS

network, verification, control plane

ACM Reference Format:
Yifan Li, Jake Jia, Xiaohe Hu, and Jun Li. 2019. Real Time Control Plane Verification. In *ACM SIGCOMM 2019 Workshop on Networking and Programming Languages (NetPL'19), August 23, 2019, Beijing, China.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3341561.3349591

1 INTRODUCTION

Datacenter networks are more likely to fail on account of configuration error than malicious attacks. Existing control plane verification tools, including ARC, Minesweeper, statically analyze network configuration, hence they need to verify the configuration all over again even if there is only a minor change. Therefore, they are not suitable for real time verification.

As data plane changes much more often than control plane, some real time data plane verification tools have already been proposed, such as Netplumber. They inspired us to design a verification layer to record the existing control plane configuration, and only verify the incremental changes of the configuration when it is modified. We implement an incremental algorithm called *the Back-Forward Algorithm* to verify the reachability based on ARC, which is the fastest existing control plane verification tool.

2 DESIGN

ARC is a control plane verification tool that works with static network configuration scripts. It converts network configuration into a weighted directed graph where vertexes are network devices and subnets and the edges are possible connections each has a capacity of 1 or inf. ARC uses the augmenting path algorithm to calculate the max flow from the source vertex S to the destination vertex T. If the max flow > 0, T is reachable from S.

The augmenting path algorithm used by ARC can be summarize as finding a path from the source vertex and the destination vertex, augmenting the flow to the capacity limit of this path, add adding an inverse path to the graph. The algorithm continues until there is no path from the source vertex to the destination vertex.

Frequent minor changes of network configuration may cause an edge added to or deleted from the graph. In this case however, ARC will have to redo a complete path augmenting all over again for even a minor configuration change. To reduce the time cost, we store the augmenting result. When an edge is added to the graph, we only need to run the augmenting algorithm from the stored result. When an edge is removed, we need to judge whether it is on the max flow path. If not, it does not matter to the reachability. Otherwise, if the removed edge is (A, C), the source vertex is S, and the destination vertex is T, we need to augment A to S, and T to C, and finally S to T. We name it as the Back-Forward Algorithm *(the BFA)*, as when an edge is removed, the algorithm augment backward twice, then forward once.

3 EVALUATION

Tested on various datesets provided by ARC Github project, the BFA reduces the verificaiton time by about 80% when an edge is add, and about 60% when an edge is removed. The detailed result is shown in Table 1.

Table 1: Augmentation Computing Time (ms)

Dataset	*Original*	*BFA Adding*	*BFA Deleting*
batfish-nsdi	29.40	4.72	9.14
bgp_length	25.94	5.48	9.77
bgpospfchain	23.75	5.58	11.97

The datasets provided by ARC are all quite small. The cost time of max flow calculating grows quadratically when the network scale grows. Therefore, the time cost reduction introduced by our proposed BFA will be much more significant in practice.