On the Cascading Failures of Multi-Controllers in Software Defined Networks

Guang Yao and Jun Bi*
Institute for Network Sciences and Cyberspace/ TNList
Tsinghua University
Beijing, China
{yaoguang@cernet, junbi@tsinghua}.edu.cn

Luyi Guo
Department of Communication Engineering
BUPT
Beijing, China
guoluyi@bupt.edu.cn

***Abstract*—In this paper, a potential threat to reliability of Software Defined Networking (SDN) is disclosed: the cascading failures of controllers. Current SDN designs have widely utilized multiple controllers and the load of a failed controller can be redistributed to the other controllers. However, simply utilizing multiple controllers cannot protect SDN networks from a single point of failure: the load of the controllers which carry the load of the failed controller can exceed the capacity of them, and then cascading failures of controllers will happen. In this article, at first we propose a model for such failures and present simulation results based on the model. Strategies for initial load balance and load redistribution after failure are designed to prevent such failures. The simulation result shows the strategies can significantly increase the resistance of SDN networks to cascading failures.**

Keywords—cascading failure; software defined network

I. INTRODUCTION

Software Defined Networking (SDN) proposes to decouple the forwarding plane and control plane of network devices, and the complex control plane functions are offloaded to the controllers of the networks. For that the complexity of networking in SDN networks is handled by the control plane, there are some challenges in designing the control plane architecture: reliability, consistency, scalability, generality, etc [1]. Considering SDN networks will lose packet forwarding function if the control plane fails, the reliability of the control plane is critical for SDN to be adopted by production networks. To prevent the controller becoming a single point of failure is one of the most important reasons for shifting SDN control plane from the one-controller model [2] to the multiple-controller model [1,3]. In a SDN network with multiple controllers, if one controller fails, other controllers can take over the network devices managed by the failed controller [1,3].

The multiple-controller model enhances the reliability of SDN networks. However, it does not guarantee the reliability of SDN networks under single point of failure. In this paper, we disclose one of the remaining threats to SDN networks with multiple controllers: cascading failures of controllers. According to the measurement of existing researches [4] and our earlier deployment, if a network uses flow-based traffic management, massive forwarding plane events must be handled by the controllers. This load is non-trivial for current typical servers and the controllers may operate with high load. As illustrated in Fig. 1, if one controller fails, the load of the controllers which take over the load of the failed controller will increase. Possibly the load will exceed the capacity of the controllers, and the controllers will turn out of service. Though the cascading failures may stop after a number of controllers fail, it is possible that all the controllers turn down at the end. As a result, the whole SDN network becomes paralyzed triggered by the failure of only one controller. To understand cascading failures of controllers and design mechanisms to prevent such failures are critical for the control plane design of SDN. In this article, we explore the cascading failures of controllers in SDN networks.

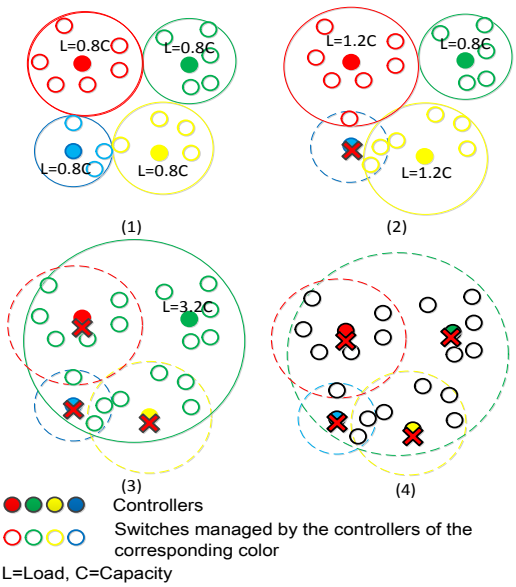


Fig. 1. A cascading failure in a SDN network:(1) the network runs normally with the load of each controller is 80% of its capacity; (2) the blue controller fails and the load is re-distributed to the red and the yellow controller, whose load are then exceeding their capacities; (3) the red and the yellow controllers fail, and the green controller takes all the load to manage the network; (4) the green controller fails at last and all the switches are out of control.

Our contributions are of three folds:

(1) We disclose the possibility of cascading failures in SDNs;

(2) We propose a model for such cascading failures;

(3) We design effective strategies to prevent such failures.

This research is supported by the China Postdoctoral Science Foundation funded project (2013M530047), the National High-tech R&D Program ("863" Program) of China (No.2013AA010605), the National Science Foundation of China (No.61073172).

* Jun Bi is the corresponding author.

II. MODEL AND SIMULATION RESULT

The model is inspired by the Motter model [5], which is a cascading failure model in complex networks. For a SDN network with N controllers, the capacity of a controller is the maximum load the controller can handle. Considering the capacity of a server is limited by its cost, it is reasonable to assume the capacity C_j of controller j is proportional to its initial load L_j,

$$C_j = (1+\alpha)L_j,\ j = 1,2,\dots,N \tag{1}$$

where the constant α is the tolerance parameter. Since the heaviest task of controllers is to manage flows, the number of flow entries that should be managed by a controller can be used as a metric of its load. When all the controllers are alive, the SDN network operates normally as long as $\alpha > 0$. But if a controller turns down, its load will be redistributed to other controllers, and the load of these controllers will increase. If the load of a controller becomes larger than its capacity, the controller will consequently fail.

Based on the simulation result illustrated in Fig. 2, it can be found that cascading failures could make the whole SDN network collapse with high probability, especially when the initial failed controller carries the maximum load. This result shows the SDN network is quite vulnerable without resolving the cascading failure risk, even when multiple controllers are deployed.

III. OPTIMAL STRATEGIES

To prevent cascading failures in SDN networks, the following requirements must be satisfied:

1. The whole system must have sufficient capacity to tolerate the load of the failed controller, i.e., the tolerance parameter

$$\alpha \geq \frac{1}{N-1} \tag{2}$$

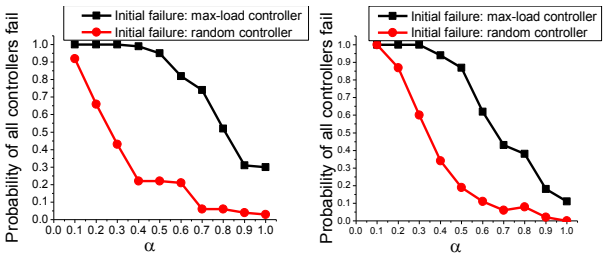
2. The initial load must be balanced among all controllers to avoid the failure of a controller that carries the maximum load turning down the whole network,

$$\max\{L_i\} \leq \frac{\alpha}{1+\alpha}\sum_{1\leq i\leq N} L_i \tag{3}$$

3. The load redistribution after a failure must not cause the load of any controller exceeding its capacity,

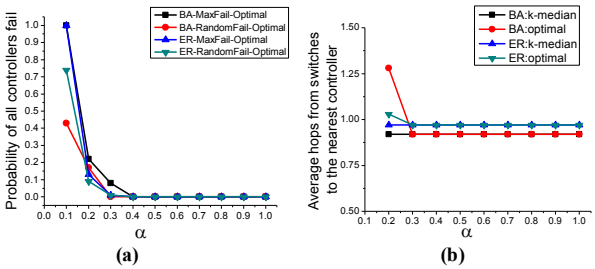
$$\frac{\Delta L_i}{L_i} \leq \alpha \tag{4}$$

The proof of these requirements is trivial. However, it is worth noting that, a controller deployment strategy and a load redistribution strategy which only optimizing the response time from each switch to the nearest controller may violate the requirements. This point is important but not exposed by existing researches. Firstly, deploying a small number of controllers is enough to achieve satisfying response time, with no tolerance of failure required. Secondly, a minority of controllers may take a large proportion of load, especially in scale-free networks. Thirdly, to achieve minimum response time, the load of the failed controller will be distributed to the nearest controllers, which may not have enough capacity to take the load.



(a)BA Network (n=100, m=5) **(b)ER Network (n=100, p=0.1)**

Fig. 2. The probability of all the controllers fail in BA(Barabási–Albert, a scale-free network model) and ER(Erdős–Rényi, a random network model) networks after: 1. the controller carrying maximum load fails; 2. a random controller fails. The controller deployment strategy is *k-median*. The number of controllers is 10. The managed switches of a failed controller are taken over by the nearest controllers alive after the failure. The average value of 100 simulations in each scenario is presented.



(a) **(b)**

Fig. 3. (a) With optimal strategies, the probability of all the controllers' failing falls greatly. (b) The load re-balance strategy does not increase the average hops from each switch to the nearest controller significantly.

The proposed optimal strategies include an initial load balance strategy, which ensures the load of controllers satisfies (3) when the tolerance parameter satisfies (2), and a load redistribution strategy, which ensures the load redistribution satisfying (4). Through comparing Fig. 3(a) with Fig. 2, it can be found that the strategies can prevent cascading failures effectively. Besides, although the load balance strategy may not produce a *k-median* deployment, the result in Fig. 3(b) implies the load re-balance strategy does not increase the response time significantly compared with *k-median* deployment strategy.

IV. CONCLUSIONS AND FUTURE WROKS

This paper disclosed a potential threat to reliability of SDN networks: the cascading failures of controllers. A model of such failures is proposed and strategies are put forward to prevent such failures. In the further studies, we will implement the strategies in controllers to better evaluate them in real networks.

REFERENCES

[1] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. ONIX: a distributed control platform for large-scale production networks. OSDI'10, pages 1-6, Berkeley, CA, USA, 2010.

[2] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: towards an operating system for networks. SIGCOMM CCR, 38(3):105-110, July 2008.

[3] A. Tootoonchian and Y. Ganjali. HyperFlow: a distributed control plane for openflow. INM/WREN'10, pages 3, Berkeley, CA, USA, 2010.

[4] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. DevoFlow: scaling flow management for high-performance networks. SIGCOMM '11, pages 254-265, New York, NY, USA, 2011.

[5] A. E. Motter and Y.-C. Lai. Cascade-based attacks on complex networks. Phys Rev E, Dec. 2002.