Huan-zhao Wang, Peng Zhang, Lei Xiong, Xin Liu, Cheng-chen Hu, 2016. A secure and high-performance multi-controller architecture for software-defined networking. *Frontiers of Information Technology & Electronic Engineering*, **17**(7):634-646
http://dx.doi.org/10.1631/FITEE.1500321

A secure and high-performance multi-controller architecture for software-defined networking

Key words: Software-defined networking (SDN), Security, Multi-controller, Distributed rule store

Corresponding author: Peng Zhang
E-mail: p-zhang@xjtu.edu.cn
ORCID: http://orcid.org/0000-0001-7721-2675

Motivation

- Software-defined networking (SDN) is a promising architecture for the next-generation Internet, and controllers play a critical role in SDN.
- Existing single controller SDN architectures are vulnerable to single-point failures, where a controller's capacity can be saturated by flooded flow requests.
- Due to the complicated interactions between applications and controllers, the flow setup latency is relatively large.
- A new SDN controller architecture that can fast respond to flow requests and tolerate single-point failures is needed.

Main idea

- The controller pre-computes flow rules and caches them across multiple controllers.
- When a flow request comes, the controller simply looks up in its cache for the corresponding rules, without the interaction of applications.
- Controllers periodically check the consistency of their rules with each other, so that when rules at one controller are maliciously modified, they can be detected and repaired.

Method

1. Partition flow rules and stores them in a distributed hash table (DHT), which is inspired by Chord (Stoica *et al.*, 2001).
2. To maintain consistency, controllers cooperatively check each rule periodically with majority voting, and Merkle hash tree (MHT).
3. We formulate the controller assignment problem as a linear optimization problem, and use a simple heuristic algorithm (Algorithm 2) to find an approximate solution.

Major results

- DRS can respond to flow requests faster than the ONOS controller

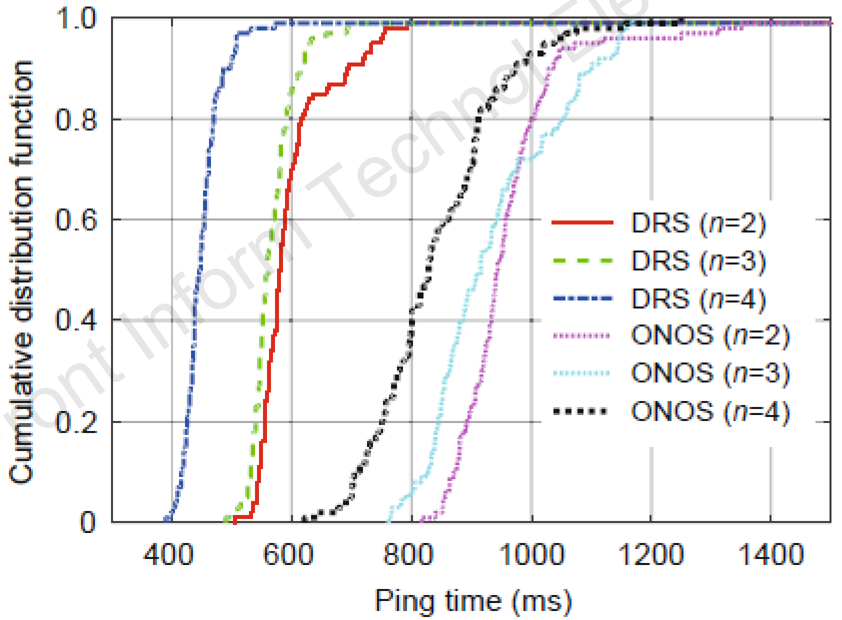
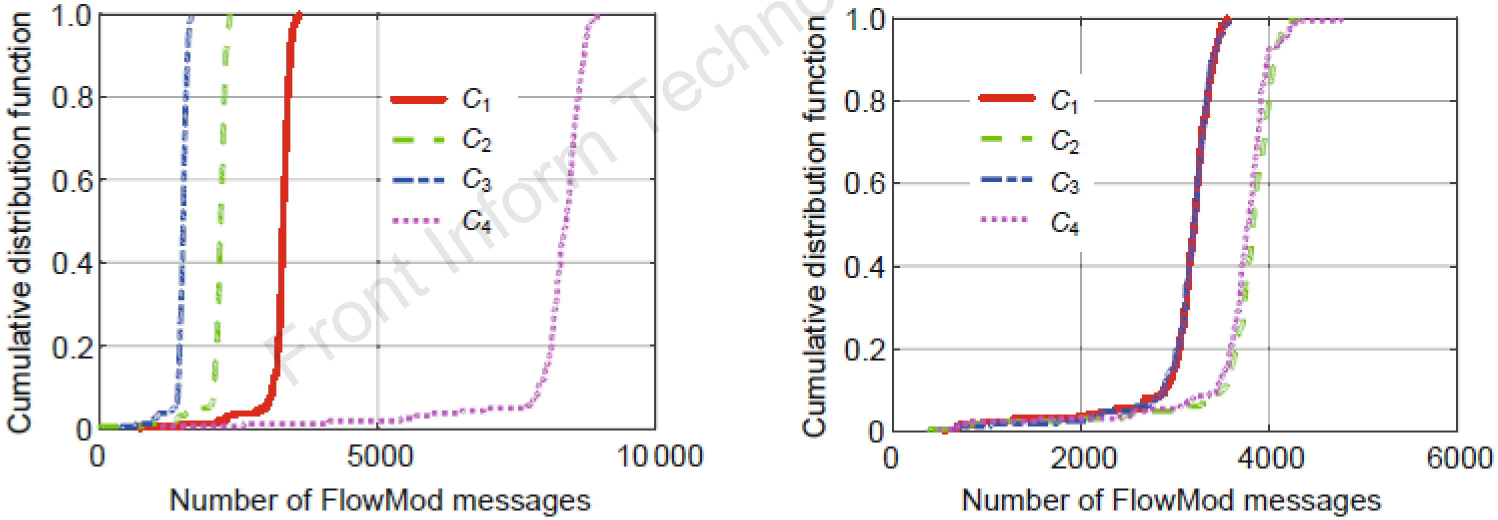


Fig. 7 **Comparison of DRS and ONOS on ping time. The ping is made from the first to the last switch in a linear topology of 100 switches**

Major results (Cont'd)

- DRS can achieve a balanced load across multiple controllers using the controller assignment algorithm



(a) (b)

Fig. 10 Processing load of each controller, with random assignment (a) and Algorithm 2 (b)

Conclusions

- DRS uses rule caching to reduce the response time to flow requests, and thereby can achieve a higher processing throughput.
- DRS uses an MHT-based consistency check to efficiently detect rule modifications.
- Experiments demonstrated the efficiency of consistency check and rule update, and also showed that DRS outperforms native Floodlight and ONOS, in terms of flow setup time and processing throughput.