Lossless Reconfiguration to Data Plane for Multi-Domain SDN

Boyang Zhou[1], Chunming Wu[1]*, Ge Pan[2], Haifeng Zhou[1], Bin Wang[1], Yansong Wang[3]
College of Computer Science, Zhejiang University, Hangzhou 310027, China[1]
Department of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China[2]
ZTE Corporation, Nanjing 210012, China[3]
E-mail: {zby,wuchunming,julius,bin_wang}@zju.edu.cn, hf.zhou.ccnu@gmail.com, wang.yansong@zte.com.cn

1 Introduction

For the multi-domain SDN, each domain connects with others via their border switches. Reconfiguring those border switches at the control plane is of extremely frequent and important to numerous network services and management tasks. Especially those large-scale reconfiguration tasks demand large numbers of involved services to reconfigure their forwarding rules of the border switches, e.g. the inter-domain traffic load-balancing task to mitigate congestion. As the reconfiguration brings many in-flight packet losses and flow connection disruptions in the process, whether it can be performed in a lossless way will directly impact the availability of these involved services, the feasibility of this type of task and even the performance of the SDN.

No existing work has exploited the advantage of SDN architecture to resolve the problem, i.e., the standard platform of controlplane being capable of managing the states of all switches in its domain. The state-of-the-art in SDN research, especially those on the current realizations of SDN controllers have not equipped with the relevant components to deal with such multi-domain problem[1][2]. Besides, the literature on the relevant problem in Internet research has some proposals[3]. However, these proposals fail to avoid the in-flight packet losses, many of them are designed for a specific type of service rather than any type, and some even have high time and communication costs, which contribute to these proposals being unpractical and cannot be adapted to the problem in SDN.

It is challenging to reconfigure the border switches without packet loss, since asynchrony of controlling the switches at controllers due to differential transmission delays and execution times in the reconfiguration process. The transient-state-induced availability problem of service is existed because of inconsistence of the states of flow matching rules at the switches in the reconfiguration process, causing packet losses and flow connection disruptions and impacting the availability of the services. We explore this problem and propose our approach which completely resolves it. First, we design a reconfiguration protocol for coordinating the controllers to reconfigure their border switches at a virtually synchronized time tick to overcome the problem. Second, we implement the protocol as a primitive which is available for any type of service or management task. Third, we design a new layer (named as supervision layer) added in the controller and implement the primitive in this layer.

In general, the proposed approach has three remarkable advantages compared with the literature proposals:

(1) **Any type of service or management task adapted**: The reconfiguration protocol is designed independently to any service or management task and is designed as a primitive in the supervision layer. Any service or management task can call this primitive. The current literature proposals can only improve the availability for a certain service rather than any, e.g. the seamless migration[3], because they fail to find an architectural solution to improve the availability of all kinds of services.

(2) **No packet loss when reconfiguring**: The reconfiguration protocol is capable of avoiding packet losses (including in-flight packet losses) by storing the in-flight packets at the switch side at the beginning of the reconfiguration process and restoring them at the end of the process. Those in-flight packets are temporarily stored at the cheap extra memories at switches in a distributed way, e.g. DRAMs and SSDs. Because these in-fight packets of a service are distributed in the links of its original route and the whole reconfiguration time is transient, we can also temporarily store these packets in the buffer of the switches like the method introduced in [4]. Besides, some of the current literature proposals optimize the order of reconfiguration steps of data plane to minimize the adverse impacts of the problem. However, the in-flight flows can also be lost, since the resources of border switches are still randomly updated without control.

(3) **Independent evolvability**: The reconfiguration protocol is encapsulated in the supervision layer and thus is independent to the other components of the controller. As a result, it can be evolved independently from the switch and the other components of the controller. Besides, it can also resolve the reconfiguration collision problem, e.g. the possible collision between the reconfigurations of inter-domain routing and firewall, while both the current SDN and Internet literature proposals fail to consider isolating forwarding rules for multiple services.

Controllers equipped with the reconfiguration protocol primitive and the supervision layer will bring the following benefits:

(1) **Reconfiguration Flexibility**: The reconfiguration protocol primitive offers interfaces for services and management tasks. As a result, they do not need to separately devise related

*corresponding author, E-mail: wuchunming@zju.edu.cn

This work is supported by the National Basic Research Program of China (973 Program) (2012CB315903), the Key Science and Technology Innovation Team Project of Zhejiang Province (2011R50010-05) and the National Natural Science Foundation of China (61379118 and 61103200). This work is sponsored by the Research Fund of ZTE Corporation, and also supported in part by the BBN/NSF Project 1783.

mechanisms to avoid packet losses and connection disruptions by themselves in the reconfiguration process. Consequently, except the reason of no packet loss, performing management tasks becomes more feasible, e.g. inter-domain load-balancing and inter-domain routing changes, by demanding involved services to call the protocol primitive in turn.

(2) **Improved availability of services in reconfiguring**: The extra memories at switches make in-flight packets be restorable. Those availability-sensitive services can be well protected from the packet losses and connection disruptions in the reconfiguration process, e.g. Voice-over-IP (VoIP), E-commerce and VPNs, while the current SDN literature proposals fail to protect the in-flight packets and also contribute to severe disorder of the following packets.

(3) **Control efficiency of multi-domain SDN services**: The reconfiguration primitive is implemented and executed from the controller to control the extra memories of switches in a distributed manner, to deal with frequent reconfiguration of data plane required by multiple services. Each service can initialize the reconfiguration protocol without any collision with other services. The control efficiency is improved by that flexibility of reconfiguration, so that much less packet losses and flow connection disruptions is produced.

2 Mechanism

The supervision layer locates between the core and the local network view layer at the controller (see Fig.1). This layer supervises the states of in-flight flows and the forwarding rules for all the border switches in the current domain. The services that run on the local view update their forwarding logics by writing its new states to the supervision layer via the view. The new states are detected for their collisions with other services according to their matching priority. The design of this layer promotes the efficiency of the reconfiguration by aggregating the recent updates and compressing them into one single reconfiguration operation.

The reconfiguration protocol is designed by extending the OpenFlow protocol for conveying commands of storing and restoring the in-flight flows. We also extend the switch architecture with a new extra memory for temporarily storing the in-flight packets by utilizing DRAM. Besides, the current industrial NEC PF5240 switch proves feasibility to store flow header states in its buffer[4], thus it is also feasible to store these packets in the buffer of the switch considering these packets being widely distributed in the links of its original route and the time of reconfiguration being transient. The protocol has three major phases that are strictly separated with each other: (i) temporarily storing the in-flight flows of all the border switches before reconfiguring; (ii) updating the forwarding rules of the border switches to the new states; and (iii) restoring the in-flight flows states of all switches after updating. The time and message complexities are in logarithm and quadratic scales with respect to the average number of border switches for each domain.

3 Performance

Our current implementation of the protocol is in the NOX controller to support services and management tasks as a standard-

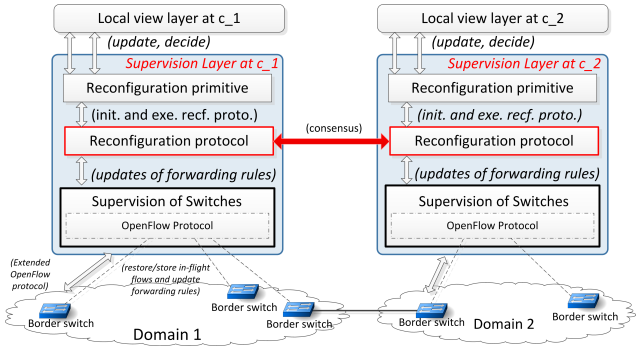


Figure 1: Architecture of Supervision Layer

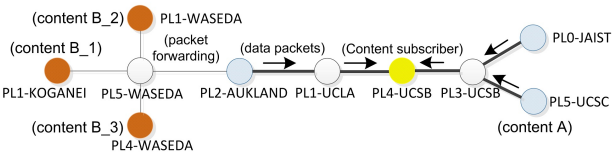


Figure 2: Emulation Topology of ICN

ized primitive. We implement a prototype of Information Centric Networking (ICN) based on CCNx realized as a SDN service to the controller together with the reconfiguration protocol. The forwarding information base (FIBs) of ICN is implemented as the flow table at the switch side. All the rest of ICN functions are implemented at the controller side, where the content routing reconfigures the FIBs in the presence of link congestions.

The effectiveness of the protocol is evaluated on the PlanetLab testbed with 9 nodes of high bandwidth (see Fig.2). Nodes are connected with each other via MST-based overlays. Fig.2 shows three blue consumers send Interest packets at 1000 per second in Poisson. Fig.3(a) shows the availability ratio (equaling to the number of Interest packets divides data packets) is averaged on 1 for our prototype, where the spur points have values larger or less than 1 (the solid red line), while the raw prototype is less than 1 that data packets are lost in updating. Fig.3(b) shows that the mean and standard deviation are 40.5% and 21.56% less than the raw implementation.

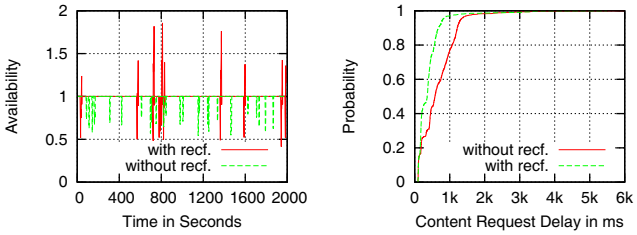
References

[1] L. Vanbever, J. Reich, T. Benson *et al.*, "Hotswap: correct and efficient controller upgrades for software-defined networks," in *ACM SIGCOMM workshop on HotSDN*, 2013.

[2] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates," in *ACM SIGCOMM workshop on HotSDN*, 2013.

[3] S. Vissicchio, L. Vanbever, C. Pelsser *et al.*, "Improving network agility with seamless bgp reconfigurations," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 990–1002, 2013.

[4] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 127–138, 2012.



(a) availability (b) request delay

Figure 3: Performance of ICN