International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 7 Issue: 6 59 - 64

Feasibility Analysis of the Algorithms: Secured and Efficient Routing Path Update in Software Defined Networking (SDN)

Mohammad Ashraful Hoque
Southeast University
Dhaka, Bangladesh
ashraful@seu.edu.bd

MD Shibli Mollah
Southeast University
Dhaka, Bangladesh
md.ta.shibli@gmail.com

MD. Razu
Southeast University
Dhaka, Bangladesh
razu@gmail.com

Abstract—Software-defined networking is the talk of the town in today's networking industry. Because of the limitations of traditional networking, SDN is getting more popular every year. Lots of researches are taking place to improve the efficiency and overcome the challenges of SDN though it has many advantages. Hence one key problem of SDN is the network update. If the route update does not perform well, it causes congestion and inconsistencies in the network system whereas bandwidth utilization and security is our main concern. We have compared two pre-built algorithms especially for routing path update and proposed a new algorithm with maximum security and loop-free network.

Index Terms—*SDN, Network Update, Algorithm, Proactive-routing, controller, Mininet, Python, OpenFlow, Network relia-bility, Programmable networks, TCAMs*

*****

I. INTRODUCTION

The Internet is an inevitable part of our daily life and the network has always been very traditional. We have specific network devices like routers, switches, and firewalls that are used for specific tasks. These devices are only operated by vendor specific commands and there is least or no programma-bility. Moreover, the major technologies, such as distributed control and transport network protocols which are run inside the routers and switches, helps these devices to transmit digital packets throughout the world. Though these technologies are adopted worldwide, traditional IP networks are complicated and difficult to manage [1]. Current IP networks are unable to manage automatic reconfiguration and response mechanisms that don't exist virtually. So, it's a great challenge to enforce the required policies to solve dynamic problems in the tradi-tional environment. Datacenter networks, enterprise networks, carrier networks etc. have become critical infrastructures in today's network industry. Today's network system is so old fashioned that the technologies and network functions used to manage, develop, debug or troubleshoot different computer networks are similar which were used in the '90s [2]. So, it fails to perform with strict requirements in terms of correctness and availability. Tech giants like Amazon, GitHub, GoDaddy etc. sporadically report issues like slow responses and intermit-tent errors with their network, due to limitations of traditional devices, misconfigurations, e.g., resulting in forwarding loops. The integration of current IP networks is vertical. Here, the control plane (that decides how to handle network traffic) and data plane (that forwards traffic according to the decisions made by the control plane are not separated and bundles inside the networking devices. It lessens flexibility and puts a barrier in innovation and evolution of the networking infrastructure [2].

Software-Defined Networking (SDN) is an emerging net-working paradigm that helps us to overcome the limitational of the existing traditional network. It is changing the whole concept of the networking infrastructure and introduce a more reliable and significant network system with less dependency on the devices. Here, the control plane and the data plane are separated which breaks the vertical integration and enable us to have a programmable and more efficient network. The SDN switches act as forwarding devices in the data plane whereas the decisions are made through the centralized controller. Moreover, the SDN architecture is comprised of three different layers, such as the Application Layer, Control Layer, and Infrastructure or Data Plane. The separation of the control and data forwarding functions is denoted as "disaggregation". This revolutionary architecture provides us with more information about the entire network system, which is software-driven and totally opposite to the traditional network system. The control plane takes the responsibility of the routing protocols, middlebox configuration, and decision making whereas the forwarding is done in the data plane, such as SDN switches. The application layer communicates with the SDN controller through northbound interfaces (NBI). So, a central controller can take control of all the devices connected

with it. Restful API'S (Application Programming Interface) such as the post or get messages and python programs are used to communicate between the planes to disseminate information. On the other hand, the SDN controller communicates with the data plane through southbound interfaces (SBI). Overall SDN plays a significant role to ensure flexibility, programmability, smooth and agile networking experience that is cost effective and easily manage complex network systems.

In this paper, we have focused on the network update in SDN. Apart from the advantages, it is still a challenge to provide consistency with zero packet loss guarantee and flow control.

II. ROUTING ALGORITHMS

SDN centralizes most of the distributed routing algorithms used in routers. SDN, however, is also a system for a single domain or collection of domains under a single network operator. Within this, the need for routers is eliminated. So, there is no such thing as routers in SDN. The Network is generally made of switches and controllers. But the routing algorithms are used in controllers and the routing table is in the control plane because it is where routing protocols such as OSPF, IBGP, BGP, EIGRP control how the protocols (ipv4 and ipv6) will be routed. The controller typically holds in memory a graph of all the switch nodes in the domain. Consequently, it does not typically need route tables. It is forwarded a packet from a switch which contains the destination, the controller searches the graph for the fastest (or cheapest, or ...) route to the destination and immediately creates and installs new switch rules in every switch en-route. Routers implement decentralized routing algorithms, that is, they talk to each other and over time converge towards the best routing path. In the event of a router failing or being added to the network, the network self-heals and again over time converges towards the best routing path. SDN implements centralized routing, that is it assumes a central controller that knows where all the switches and end hosts are and can map the shortest path across the network. It will then install rules on the switches involved that allow flows to traverse that path without further contact with the controller (the controller typically sees the first packet). Several algorithms have already been developed to reduce latency and balancing the traffic load of the networks [5]. The first algorithm, named shortest path first (SPF) algorithm, will focus on finding the shortest path from source to destination. In order to utilize the network bandwidth, if more than one shortest path exists, we will select the path, which has the maximum bottleneck bandwidth (MBB). SPF algorithm is used to route the latency sensitive applications. The second algorithm, named bandwidth-aware routing (BAR) algorithm focuses on finding a path with MBB from source to destination. If more than one path with the same MBB exists, we will select the one which has the shortest path. BAR algorithm is used to balance the workload of the switch links. The third algorithm is the k-SPF, which can find a path with MBB among the first k shortest paths. The fourth algorithm is K-bar which finds the shortest path among the first k MBB paths on a network. One significant challenge faced by SDN is the communication channel between the logically centralized control platform and the SDN switches in the data plane. When an update occurs, network update commands (e.g., OpenFlow FlowMod messages) may lead to transient inconsistencies such as loops or bypassed waypoints (e.g., firewalls) [6]. One approach to ensure transient consis-tency even in asynchronous environments is to employ smart scheduling algorithms. WAYUP [7] and PEACOCK [8] are two update scheduling algorithms to ensure reliable network updates. WAYUP [7] is based on security purpose whereas PEACOCK [8] is implemented to ensure loop-free networks in SDN.

III. RELATED WORK

SDN was coined in 2005 by Martin Casado who developed and implemented his idea at Stanford University. So, several types of research have already been done to create more functionality, efficient algorithms to make a viable network within thirteen years of journey. We have found many great ideas about routing path updates and this section gives a brief description to them.

The first anticipation of SDN update is represented by Reit-blatt et al. in [13], [14] and it is the first analysis of security is-sue in any perspective of SDN updates and ensure consistency in forwarding policy. It mostly focuses on a change in network configuration. They identify two distinct consistency levels, per-packet and per-flow, and present general mechanisms for implementing them in SDN. When there is a change in the configuration the controller attaches packet with the version of configuration, they forward at the ingress switches. While updating the switches with new rules, it also keeps the old rules. It also added more information in the and modified header packets of the new rules. It only deletes the old rules when it gets a confirmation that the new rule is updated.

Moreover, the new and old rules already occupy space in the memory locations which consumes Tertiary Content Addressable Memory (TCAM) used as a memory resource and it is very expensive. Though it ensures per-packet and per-flow consistency, here memory resource is not utilized, and more works are needed to be done.

Several applications like load-balancing and failure recovery are developed for SDN so that the controller can take decisions by updating flow tables in data plane to avoid network conges-tion and failure. In [19], their concentration area is preserving throughputs of flows. A heuristic

dependency algorithm is implemented to reduce the flow table space overhead when an update may occur. Finally, Cupid [19] can update faster with fewer throughput losses in fat-tree and mesh topologies but security issues are not discussed here.

The fundamental problem in SDN route update is under the dynamic condition when routes change, it becomes indeterminate. Rick McGeer [16] proposes a safe and efficient update protocol for OpenFlow networks. When there is a change in the ruleset the controller computes which packets are affected by the update. Then the switch sends the affected packets to the controller before and after the update takes place. When the update concludes, the controller sends the packets to the network. This technique puts both rulesets on a switch, as a result, the controller consumes more bandwidth and it's not utilized.

As SDN is a distributed network system, flow table update consistency is a significant issue. Likewise, it may experience different transmission delays from controller to different network devices and the processing time of the control message from the device to device is not the same as well. So, switch update sequence is proposed in [17] as a key factor. Switches affected by the update procedure are divided into different categories depending on their forwarding behavior. The overall update procedure is comprised of three phases as the preparation phase, path reconfiguration phase, and feedback phase. Low-complexity, low-cost mechanism and potential conflict are avoided in update process. Finally, it avoids looping problem, less updating duration and packet loss is achieved.

A new approach to the problem of consistent multi-switch updates for SDN is proposed in [18]. When an update occurs, the controller computes both the old and new routes. After that, a packet is sent through both routes and ensures when the packet successfully reaches the destination. This means that both routes are working fine, and the controller replaces the old routes with the new routes in the flow table. This multicast-based algorithm ensures the connection operational throughout the change, without packet-loss. But the limitation of it is that the packets are sent through both old and new routes which consume more bandwidth and per-packet consistency is not maintained.

Consistent migration of flows is another area to work with SDN. One key problem studied in [19], is providing increased bandwidth to an application while keeping all other flows in the network and migrating them consistently to the other paths. They decided that consistent migration is possible in polynomial time and it also remains consistent if a new flow is inserted into the network. But there is no step taken for security issues.

IV. PLANNED ROUTE UPDATE IN SDN

A. WayUp Planed Route Update

WayUP is a security-based planed route update algorithm and a schedule-based routing algorithm which ensures the security whereas the main focus is the waypoint. Here, the waypoint is a firewall which acts as a filter point wherever every packet is forced to go through it. If the routing path needs to update, it updates the switches which are present on the new route but not on the old route. Then it updates the switches which are on the old route from waypoint node to the destination. It also updates the existing switches in reverse flows while new route arrives from source to the waypoint node. This algorithm was proposed by Ludwig [7].

B. Peacock Planed Route Update

Our work is based on planned route update in SDN. Over the years, plenty of works are done in SDN and the most focused area is network update related issues such as security, congestion, black hole, forwarding loop, loop-freedom etc. In [8] authors propose an algorithm named Peacock which can prevent looping problem during update in the network. Peacock algorithm is the most likely scheduled-based algorithm which works in two phases. One phase is called Shortcut and another one is Prune phase which is as follows:

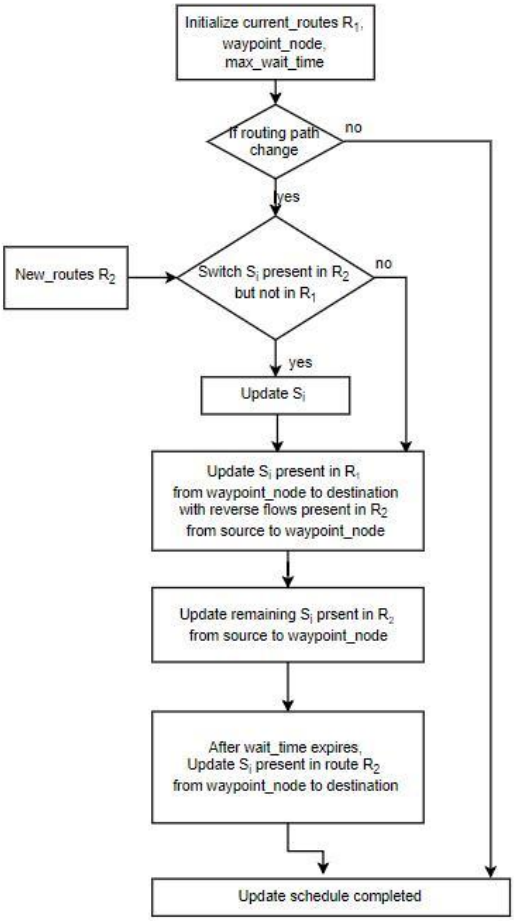

Fig. 1. Flow Chart of WayUp

1) Shortcut: From starting node to ending node peacock makes a short distance by updating the disjoint set of distant reachable forwarding edges in the new route. It defines the distant reachable forwarding edges, distance skips the nodes in old route from source to destination. By updating distant forwarding edge, it obtains many branches of the tree, where one of them contain source to destination path and this update occurs in odd rounds (i.e. 1st, 3rd, 5th etc.).

2) Prune: In this phase peacock updates all nodes, those are in new route from source to destination. Since in Shortcut Peacock updates distant forwarding edges in Prune phase, merging those nodes and reduce the distance, and re-establish the line again so that, based on that line it can define next disjoint set of distant updatable edge. This update occurs in even round (i.e. 2nd, 4th, 6th etc.). This two-phase continued until all nodes are updated in the new route.

RESULTS

C. Update Duration of WayUp

We have found that the wayup update duration increases with the number of modified flows but not exponentially which is a positive approach to this update method. The result of wayup update duration is shown in Fig. 3

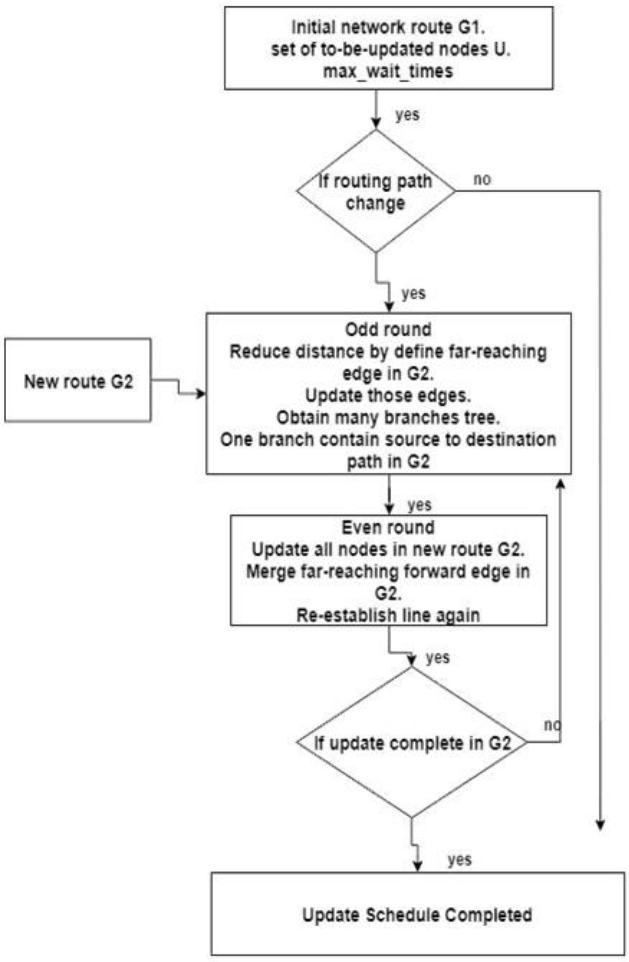

Fig. 2. Flow Chart of Peacock

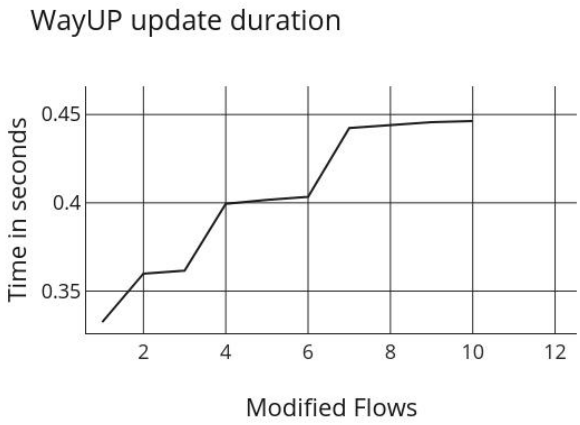

Fig. 3. Update duration of WayUp

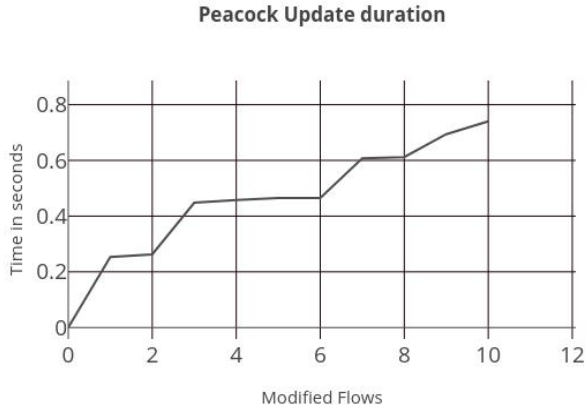

Fig. 4. Update duration of Peacock

the modified rules. The result shows us that the duration of update increases according to the number of modified flows. But update duration does not increase exponentially which is the main advantage of peacock update technique.

V. BANDWIDTH UTILIZATION A. TCP Throughput of WayUp

We have successfully performed TCP operation in WayUp. We have used mininet and iperf command to perform the operation to generate this TCP throughput. In this scenario we have used twelve switches and two hosts: one host acts as a TCP client and another one acts as a server. In Fig.5 X-axis displays the operation time and Y-axis displays the bandwidth. We set the bandwidth limit to 10 Mbps for this test. Finally, we have found the average TCP throughput but after a few times, it has lost its routing. So, there is a communication gap between the host server and client during this time.

D. Update Duration of Peacock

We have demonstrated the result of peacock modified flows update duration in X-axis. Here, we have shown ten modified flow entries and time in seconds in Y-axis. Fig. 4. displays the

number of flows against update duration. We have observed that update duration of Peacock that increases additively with

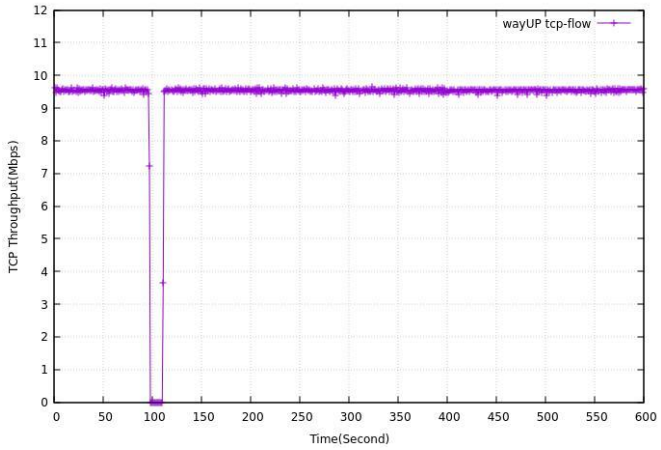

Fig. 5. WayUp tcp Throughput

B. TCP Throughput of Peacock

We have performed the same TCP operation in Peacock and the same configuration is used to perform the TCP operation. In Fig.6 X-axis displays the operation time and Y-axis displays the bandwidth. The bandwidth limit remains the same as 10Mbps. Our observation for Peacock: It's throughput is dropped a few seconds later than WayUp.

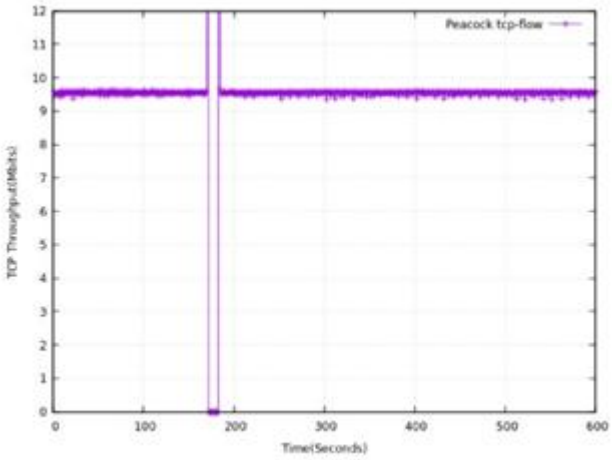

Fig. 6. Peacock tcp Throughput

VI. UDP THROUGHPUT

A. WayUp UDP Throughput

We have performed the same operation for UDP in WayUp and the result is shown in Fig.7. where X-axis displays the operation time and Y-axis displays the bandwidth. Bandwidth is also limited to 10 Mbps. We have observed that the throughput is dropped a few minutes later like we had in TCP. So, there are some similarities between UDP and TCP in WayUp.

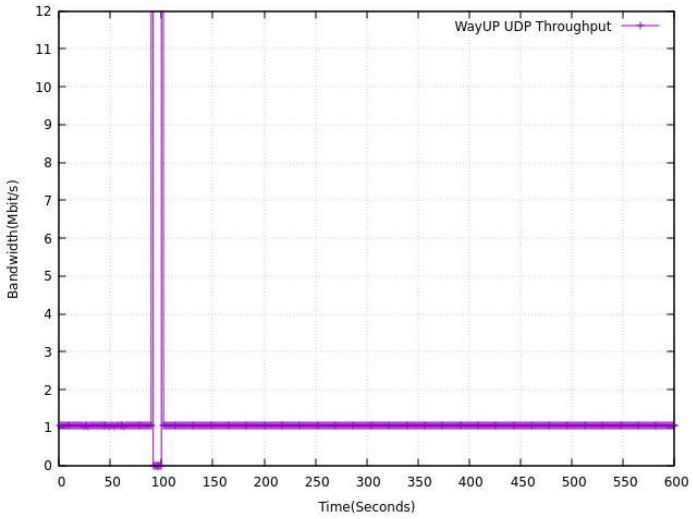

Fig. 7. WayUp UDP Throughput

B. Peacock UDP Throughput

The same operation is performed with the existing configuration for UDP in Peacock and the result is shown in Fig.8. Here we have seen a significant change, that is: the throughput isn't dropped unlikely the previous results we have found so far.

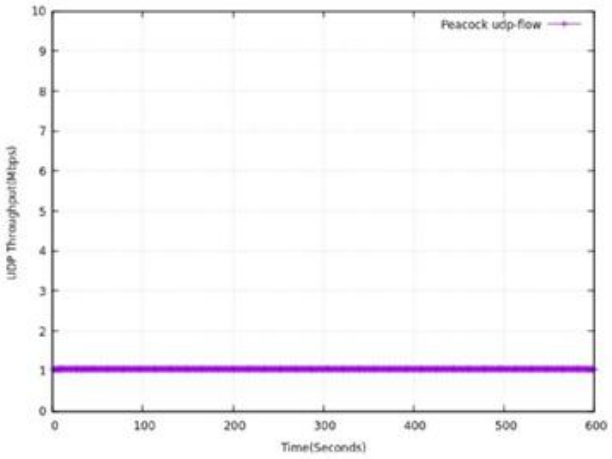

Fig. 8. Peacock UDP Throughput

VII. CONCLUSION

We have successfully implemented the algorithms to achieve less update duration, ensured security and created a loop-free network. But those are done in different algorithms in different schemes. Our target is to combine these algorithms to create a more efficient algorithm where bandwidth utilization and security will be emphasized simultaneously.

We have studied loop-free update by using peacock and ensured security by using WayUp algorithm and waypoint enforcement with minimum rounds in SDN route update. We believe that the results we have found will show a new direction in future research works. Whereas, some of the key challenges faced by SDN have already been solved and new challenges are coming.

REFERENCES

[1] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09, Berkeley, CA, USA, 2009, pp. 335–348.

[2] Kreutz, Diego, et al. Software-defined networking: A comprehensive survey, Proceedings of the IEEE 103.1 (2015): 14-76.

[3] N. Mckeown, "How SDN will Shape Networking," October 2011. [Online]. Available: https://www.youtube.com/watch?v=c9-K5O qYgA

[4] Ahmed, J. Blech, M. Gregory, and H. Schmidt, "Software Defined Networks in Industrial Automation," JSAN, vol. 7, no. 3, p. 33, 2018.

[5] J.-P. Sheu, Q.-X. Zeng, R. B. Jagadeesha, and Y.-C. Chang, "Efficient unicast routing algorithms in Software-Defined Networking," in The dawn of 5G: EuCNC : European Conference on Networks and Commu-nications, Athens, Greece, June 27-30, 2016, Athens, Greece, 2016, pp. 377–381.

[6] Shukla et al., "Towards Transiently Secure Updates in Asynchronous SDNs," in Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference - SIGCOMM '16, Florianopolis, Brazil, 2016, pp. 597–598.

[7] Ludwig, M. Rost, D. Foucard, and S. Schmid, "Good Network Updates for Bad Packets," in Proceedings of the 13th ACM Workshop on Hot Topics in Networks - HotNets-XIII, Los Angeles, CA, USA, 2014, pp. 1–7.

[8] Ludwig, J. Marcinkowski, and S. Schmid, "Scheduling Loop-free Network Updates," in Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing - PODC '15, Donostia-San Sebastian, Spain, 2015, pp. 13–22.

[9] K.-T. Foerster, S. Schmid, and S. Vissicchio, "Survey of Consistent Software-Defined Network Updates," 08-Sep-16. [Online] Available: http://arxiv.org/pdf/1609.02305v2.

[10] Markopoulou, Athina, et al. Characterization of failures in an IP back-bone. INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies. Vol. 4. IEEE, 2004.

[11] Mr. Songtao Wang, Prof. Dan Li, Prof. Shutao Xia,"The Problems and Solutions of Network Update in SDN: A Survey" IEEE Conference on Computer Communications workshops (INFOCOM WKSHPS), 2015: April 26, 2015 - May 1, 2015, Hong Kong. Piscataway, NJ: IEEE, 2015.

[12] Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed internet routing convergence," IEEE/ACM Trans. Netw., vol. 9, pp. 293–306, June 2001.

[13] Mark Reitblatt et al., Abstractions for Network Update: Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication. New York, NY: ACM, 2012.

[14] M. Reitblatt, N. Foster, J. Rexford, and D. Walker. Consistent updates for software-defined networks: Change you can believe in! In Proc. ACM HotNets, 2011.

[15] Dr. Wen Wang, Ms. Wenbo He, Prof. Jinshu Su, Mr. YiXin Chen, Cupid: Congestion-free Consistent Data Plane Update in Software Defined Networks: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications took place 10-14 April 2016 in San Francisco, CA, USA. Piscataway, NJ: IEEE, 2016.

[16] Wang, Q. Qi, J. Gong, and J. Liao, "Mitigating the Oscillations Between Service Routing and SDN Traffic Engineering," IEEE Systems Journal, vol. 12, no. 4, pp. 3426–3437, 2018.

[17] McGeer, Rick. "A safe, efficient update protocol for OpenFlow net-works", Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012.

[18] Delaet, Sylvie, et al. Seamless SDN Route Updates. Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on. IEEE, 2015.

[19] S. Brandt, K.-T. Forster, and R. Wattenhofer, "On consistent migration of flows in SDNs," in IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications took place 10-14 April 2016 in San Francisco, CA, USA, San Francisco, CA, 2016, pp. 1–9.