The Problems and Solutions of Network Update in SDN: A Survey

Songtao Wang
Department of Computer Science and Technology
Tsinghua University
Email: wangst12@mails.tsinghua.edu.cn

Dan Li
Department of Computer Science and Technology
Tsinghua University
Email: tolidan@tsinghua.edu.cn

Shutao Xia
Department of Computer Science and Technology
Tsinghua University
Email: xiast@sz.tsinghua.edu.cn

***Abstract*—Network is dynamic and requires update in the operation. However, many confusions and problems can be caused by careless schedule in the update process. Although the problem is noticed for long in traditional networks, software defined networking (SDN) brings new opportunities and solutions to this problem by the separation of control and data plane, as well as the centralized control. This paper makes a survey on the problems caused by network update, including forwarding loop, forwarding black hole, link congestion, network policy violation, etc., as well as the state-of-the-art solutions to these problems in the SDN paradigm.**

I. Introduction

Networks are not working statically. In order to keep network in a correct and efficient state, network operators frequently adjust link weights, change traffic engineering (TE) schemes, migrate virtual machines and update routing policies, etc. These updates require careful arrangement. Otherwise, problems like forwarding loops, forwarding black holes, link congestions and network policy violations will show up during the updating process, even if the network behaves well before and after update. The impact of network update is not trivial. The figure shows that about 20% network faults come from carelessly planned maintenance [1]. Moreover, the network updating period is not short. Previous work showed that 30% packets suffer from loss for more than 2 minutes after BGP updating[2]. Ignoring the problems during the network update process is indispensable for network operators because customers' demands become more and more critical. Delay sensitive applications like financial trade, online shopping and searching cannot tolerate any network fault that affects the quality of service (QoS) of the network. For instances, Amazon would loss 1% amount of sales for every 100ms latency[3]; while Google reported that 20% traffic would drop for 500ms more search page response time.

In traditional networks, many research efforts are spent on solving the network update problem. A straightforward idea is to reduce the convergence time of a protocol[4], [5]. But it is quite difficult to design a routing protocol with fast-enough convergent time considering the size of the Internet. Further, the innovation of new protocols should not be limited by the convergent time restriction[6]. As a result, many recent works turn attention to how to eliminate the problems during the updating process on plan, instead of just mitigating them [7], [8], [9], [10]. However, it is also quite challenging to solve the problem in traditional networks. Greenburg et al. argued that in traditional networks, the forwarding decisions are made in a distributed manner and besides forwarding function the data plane also undertakes functions like "tunneling, access control, address translation and queuing" [11]. From the perspective of network update, the updating schedule produced in a distributed way can only generate locally optimized solutions and incoordination between routers make update full of faults. A rich number of management functions make it even worse because not only routing but also access control and address translation are influenced by update.

At present more and more network operators are accepting software defined network (SDN) to manage their networks. For instances, Stanford university has deployed SDN in the campus[12]. Google, Microsoft and a growing number of enterprise operators are deploying SDN in their data center networks respectively[13][14]. SDN enables new solutions to the network update problem. For example, in SDN the data plane only needs to forward packets and all the decisions like routing, load balancing and traffic engineering are made in a logically centralized controller. In Openflow, the control plane can configure data plane at flow granularity which is more precise and flexible than IP prefix matching in traditional networks. The control plane keeps a global view of the network and make comprehensive forwarding decisions for each flow. SDN operators are thus able to update the network in a finer grained manner and provide high level schedules that update the whole network. It is worth noting that SDN only makes the network update easier but do not directly offer the update mechanism. SDN switches still forward packets based on their own forwarding tables. Operators need to carefully design the update mechanisms and distribute the rules to every SDN switch.

Catching up with the trend of SDN, there is noticed growing interest in the community to design new solutions to the network update problem. In this paper we summarily describe the problems caused by network update as well as the solutions in a SDN paradigm. We hope this paper can help readers have a more clear view of this problem and foster more research on this issue. The rest of this paper is organized as follows. Section II discusses the problems caused by network updating. Section III presents the current solution to SDN update in the literature. Section IV discuss more issues in this problem. Section V concludes the paper.

978-1-4673-7131-5/15/$31.00 ©2015 IEEE

II. Problems of Network Update

Network update problem refers to network confusions caused by careless updating schemes. We assume that network states before update are correct. Network update we are talking about is not a mechanism that correct network behavior. Actually there have been a bunch of solutions for network static state verification[15], [16], [17], [18]. We do not want the update process to interrupt the network performance. Therefore in this paper we accept this principle that confusions will not occur until the start of update. Some network update scenarios and confusions that may occur are listed in TABLE I.

TABLE I. Some scenarios of network update and corresponding confusions

Scenario	Confusion
Update forwarding policy	forwarding loop, forwarding black hole, link congestion
Change access list of a firewall	network policy violation
VM migration in Data center	forwarding black hole, link congestion
Adjust traffic engineering scheme	link congestion
Update forwarding policy	network policy violation

It is hard to summarize all the network update scenarios in real networks. So we describe this problem according to the type of confusion. We pick out four basic confusions and make a description for each of them below. Many confusions in real scenarios can be regarded as a combination of the basic confusions.

A. Forwarding Black Hole

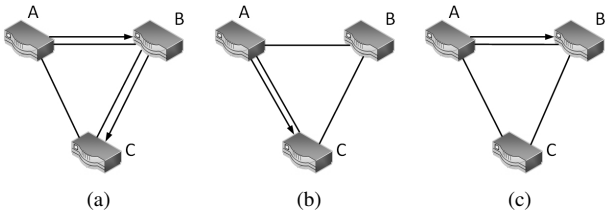


Fig. 1. An example of forwarding black hole confusion during network update.

The forwarding black hole confusion refers to the case that a packet entering an SDN switch cannot match any rule in the forwarding table during the network updating process. We illustrate this confusion by an example in Fig. 1. The update scheme is to change the path from node A to node C. The path in Fig.1(a) is the initial state and the path in Fig.1(b) is the final state. The controller's instruction is that: node A replaces an old forwarding rule with a new one and node B deletes the old rule. Since the reaching times for commands from the controller to node A and node B are different, a case in Fig.1(c) may occur. In this case node B deletes its rule first but node A has not replaced the rule. Packets arrive at node B will be sent to controller or in worse case will be discarded by node B. Node B is a forwarding black hole for packets whose destination is node C until A has updated its forwarding table.

B. Forwarding Loop

The forwarding loop confusion refers to the case that a packet suffers from forwarding loops and cannot be delivered to its destination during network update. We illustrate this confusion by an example in Fig.2. There are four switches named A, B, C and D. The arrow lines compose a tree rooted at D which represent the routing paths from the other three nodes. The update scheme is to convert the tree in Fig.2.(a) to Fig.2.(b). Node C has changed its forwarding table in Fig.2.(c) but neither A or B finish this. Thus a loop appears which means that during this period network is inconsistent. After that in Fig.2.(d), node B updates its forwarding table but A does not. The bad loop does not vanish until node A finishes updating. C-B-A is the worst update scheme in this case. Because of the latency between control plane to forwarding plane varies respect to different switches, any updating sequence may be possible if the control plane distribute update schemes to A, B and C at the same time. For the same reason it is impossible to update all the switches in an instance.

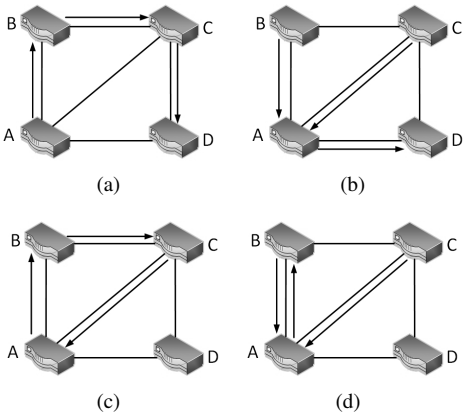


Fig. 2. An example of forwarding loop confusion during network update.

C. Link Congestion

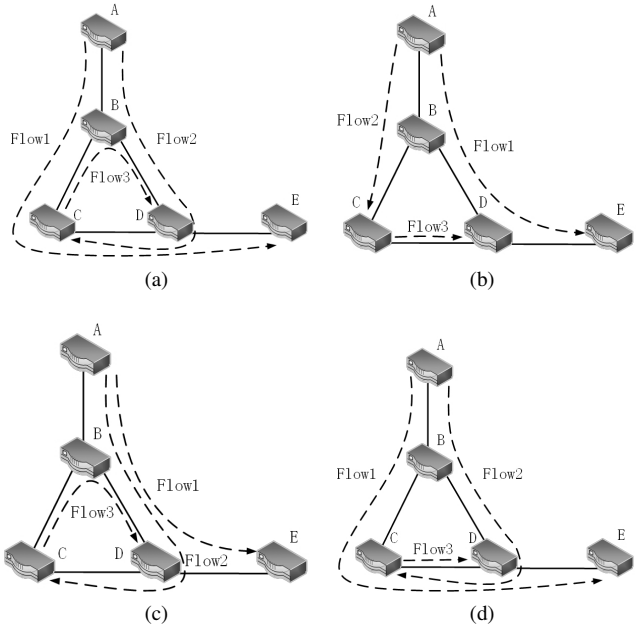


Fig. 3. An example of link congestion confusion during network update.

An example of the link congestion confusion is shown in Fig.3. There are three flows in the network. Links are

bidirectional. We assume that the capacity of each link for single direction is 10 units and the sizes of Flow1, Flow2 and Flow3 are all 5 units. A network administrator wants to change the traffic pattern from Fig.3(a) to Fig.3(b). A careless update scheme is to update Flow1 first. The result is shown in Fig.3(c). A congestion takes place on link BD which bears 15 units of traffic. As a contrary, Fig.3(d) is a good update scheme which put Flow3 as the first update flow. In this case no congestion will take place.

D. Network Policy Violation

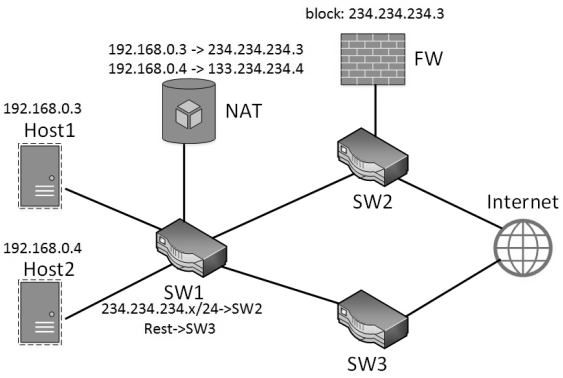


Fig. 4. An example of network policy violation confusion during network update.

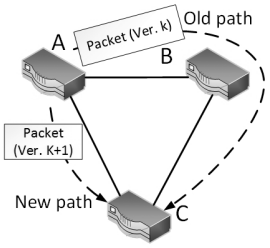
Middleboxes are widely deployed in modern networks. For example, an enterprise network consisted of 900 routers is equipped with 636 middleboxes[19]. Packets have to follow network policies which enforce packets going through a list of middleboxes (e.g., Firewall+NAT+Proxy). During the network policy update, packets may violate the policies. We illustrate this confusion with the help of an example in Fig. 4. Policy of the network shown in Fig. 4 is that all the traffic should go through a network address translator (NAT) which converts private IP address to public IP address in the packet header and only the traffic from prefix 234.234.234.x/24 should be inspected by a firewall (FW) which blocks the Internet visit from Host1. According to the policy, traffic from Host1 goes through the path of "SW1-NAT-SW1-SW2-FW" and traffic from Host2 goes through the path of "SW1-NAT-SW1-SW3-Internet". For some reasons, a network administrator changes IP mapping in the NAT. He maps 192.168.3.3 to 234.234.234.4 and 192.168.3.4 to 234.234.234.5. Thus traffic from Host1 and Host2 both go through the path of "SW1-NAT-SW1-SW2-FW-Internet". As a result, Host1 can contact the Internet now, which violates the network policy; meanwhile, traffic from Host2 also have to go through FW which increases the load of FW.

III. SDN Solutions to Network Update

In this section we firstly illustrate solutions to the four basic confusions discussed above. After that we will talk about two limitations that will affect performance of network update scheduling.

A. Solution to Forwarding Black Hole

No forwarding rule matching a packet is the reason for forwarding black hole. R. Mahajan and R. Wattenhofer describe a solution to this confusion as "add before remove" in [20]. It means that switches should update new rules first before removing old rules. "Add before remove" operation on a single switch could ensure forwarding black-hole free. Reitblatt et al. implement "add before remove" on the whole network[21]. We illustrate the solution of [21] by an example in Fig.5. Each packet is stamped with a version number k and forwarded through the network. When the update process starts, new rules with version number k+1 are distributed to switches and the switches still keep old rules with version number k. After confirming that all switches have received the new rules, controller informs switches of stamping new packets injecting into the network with new version number k+1. Then all switches wait for a time during which all packets with version number k should have left the network. After that switches could remove old rules. Because switches keep both old and new rules in forwarding tables during update process, there will be no forwarding black holes.



Old rule(Ver. k) in A		New rule(Ver. k+1) in A	
Destination	Next hop	Destination	Next hop
C	B	C	C

Old rule(Ver. k) in B		New rule(Ver. k+1) in B	
Destination	Next hop	Destination	Next hop
C	C	C	None

Fig. 5. An example solution to forwarding black hole confusion during network update.

Actually Reitblatt's solution is a stricter "add before remove". "Add before remove" operation on a single switch is enough to ensure forwarding black hole free. Reitblatt's solution enforces all switches in the network adding new rules before removing old rules. This creates a new consistent property called "per-packet consistency". "Per-packet consistency" means that a packet going through the network according to either old rules or new rules but never a mix of them. This is a stronger consistent property than forwarding black hole free. We call it 'stronger' because a "Per-packet consistency" network update scheduling can keep more consistent properties than forwarding black hole free. For example if either old rules or new rules will not create a forwarding loop, a "per-packet consistency" update scheduling also ensure forwarding loop free property. We will talk about this in next paragraph.

B. Solution to Forwarding Loop

We have mentioned that "per-packet consistency" by Reitblatt's solution[21] is a strong consistent property that also holds forwarding loop free. If we take update scheduling in Fig.2 as an example, in Reitblatt's solution the data plane keeps both the forwarding rules shown in Fig.2(a) and the forwarding rules shown in Fig.2(b) at the same time during update. Thus a packet is either forwarded along the path shown in Fig.2(a) or the path shown in Fig.2(b) but never a mix of the them.

But R.Mahajan and R.Wattenhofer point out that Reitblatt's solution is too strong for forwarding loop free property[20].

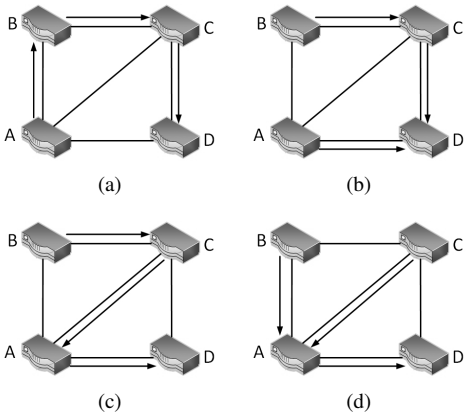


Fig. 6. An example solution to forwarding loop confusion during network update.

They show that a careful 'mix' of the old rule and new rule could hold forwarding-loop free property. With the help of the example in Fig.6, we illustrate R. Mahajan and R. Wattenhofer's solution to forwarding loop confusion. Fig.6(a) shows the forwarding path to node D before update and Fig.6(d) shows the forwarding path to node D after update. Subfigures (a)(b)(c)(d) in sequence indicate the update schedule according to R. Mahajan and R. Wattenhofer's solution. The update sequence is node A-node C-node B. This update schedule is based on a dependency tree. The dependency tree is constructed based on the forwarding path to node D after update shown in Fig.6(d) according to principles: destination node D is root of the dependency tree; node A is the child of node D and node B and node C are children of node A because in Fig.6(d) node D is the next hop for node A and node A is the next hop for node B and C. After building up the dependency tree, update schedule is arranged as: start update process from the root and then update children of each node in sequence.

C. Solution to Link Congestion

Microsoft demonstrates their inter-data center network solution: SWAN[14] in 2013. The main purpose of SWAN is to highly utilize the network capacity even if the traffic volume varies hugely by time. SWAN leaves 'scratch capacity' on links. 'Scratch capacity' will not be used until updating. [14] proves that when setting 'scratch capacity' as s, update could be finished in 1/s-1 steps without congestion. Idea of this mechanism is similar to that of [22]. They both trade updating time with scarce resource which in [14] is link capacity and in [22] is TCAM. zUpdate [23] by Hongqiang Harry Liu et al. aims to eliminate congestion in data center updating progress. They point out that in data centers switches utilize ECMP(Equal-CostMultipathRouting) to split traffic evenly among all next hops to make fully use of the redundant pathes[23]. If one link fails, traffic on this link is evenly split to the remaining links which may cause congestion. zUpdate breaks the fairness between multiple paths and split effected traffic on each path carefully when failure happens. Like [21], optimization programming model in zUpdate only requires operator providing final configuration without pay attention to update details.

D. Solution to Network Policy Violation

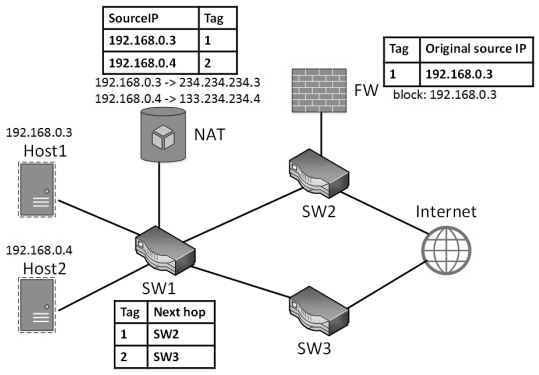


Fig. 7. An example solution to network policy violation confusion during network update.

The origin of network policy violation is that a packet cannot be matched with its original address. S.Fayazbakhsh et al. point out that the key to solve network policy violation confusion is "OriginBinding"[24]. They design a "FlowTags" mechanism for "OriginBinding" as shown in Fig.7. The NAT adds tags to packets and sends the mapping between original IP and tag to controller. Switches forward packets according to tags in packet header. The FW consumes tags. They compare tags with the mapping received from the controller and could find out the original IP of packets. No matter how many times a packet have been edited, middleboxes and switches could always map it with its original IP. For middlebox update process, a middlebox can change its configuration rules without worrying about the impact to downstream middleboxes. Network policy always hold for every packet.

E. Constraints of Network Update Scheduling

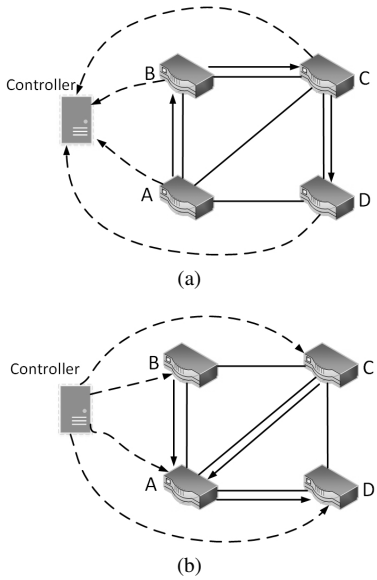


Fig. 8. Mechanism of McGeer's network update schedule.

1) Memory Constraint: Reitblatt's solution is an effective but inefficient updating mechanism which consumes 50% more TCAM capacity. Therefore some researchers try to release

the over consumed memory. McGeer[25] tries to use SDN controller as a cache during update process. Packets been affected by the update process are all sent to controller. And after update process these packets are sent back to network for further forwarding. His solution consumes no additional memory but leaving pressure to controller and the channel between controller and switches. During update, huge amount of traffic will congest the precious channel between controller and switches (eg. Openflow channel). This is a disaster for network management for the reason that fraction of configuration command may get lost. McGeer admits that his solution can only be deployed in very special cases. Katta's solution[22] is to incrementally update forwarding entries in switches. In each round a fraction of rules are updated. The memory overhead is tunable. The less memory overhead the more rounds to update. If the traffic pattern follows Zipfs law (an exponentially decaying distribution), then "80% traffic could be updated in first round and 99% traffic could be updated in three rounds which is acceptable in real network environment[22]."

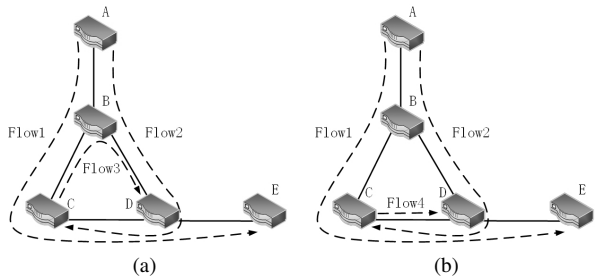


Fig. 9. A scenario of network update.

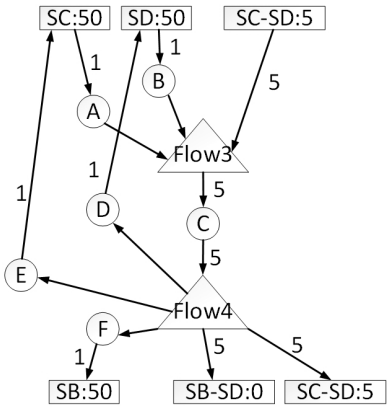


Fig. 10. Dependency graph of Fig.9.

TABLE II. OPERATIONS IN FIG.9.

Index	Operation
A	Add Flow4 at Node C
B	Add Flow4 at Node D
C	Change weight at Node C
D	Delete Flow3 at Node D
E	Delete Flow3 at Node C
F	Delete Flow3 at Node B

2) Update Efficiency: Xin Jin et al. find that the updating efficiency is quite low[26]. They find a way to speed up this process of keeping link congestion free. Fig.9 is an updating scheme example. And Fig.10 is dependency graph of this updating scheme. Rectangle represents available resource. For example 'SD:50' means 50 unit of space is available in node D and 'SC-SD:5' means 5 unit of space is available on link between node C and node D. Circle represents operation. For example operation A means add rule for Flow4 at node C. Triangle represents the path that operated by operations in circles. Arrow line and the number next to it represents resource consumed or released by the operation. For example arrow line between 'SC:50' and operation A with number '1' means operation A consumes 1 unit of memory on SC. Arrow line between operation F and 'SB:50' with number '1' means operation 'F' will release 1 unit of resource to SB. Such a model list all resources remained and transformation of resources through operations. Xin Jin et al. develop an algorithm to find a tree through all operations which obey resource limit. Resulting sequence of operators along this tree is the solution to this updating problem.

IV. DISCUSSION

R.Mahajan and R.Wattenhofer have revealed the relationship between the strength of consistent properties and dependency structure in their work [20]. Their conclusion is that stronger consistent properties require broader dependency structures to hold consistency. We borrow the idea of [20] and draw Table.III filled with consistent properties and solutions that are introduced in this paper. First column of Table.III lists consistent properties in the order strength. First row of Table.III lists dependency structure in the order of scope. A stronger consistent property requires a high level dependency structure. For example, keeping forwarding black hole free only requires "add before remove" operation on every single switch or router but keeping network policy violation free requires coordination among all network facilities including switches, routers and middleboxes. Impossible in the table means that a consistent property can not be kept by a low level dependency structure which has been proved in [20]. For example keeping forwarding loop free is impossible if update process on a single switch is carefully arranged. It at least requires the coordination of all switches which are on the downstream pathes of impacted packets.

TABLE III. NETWORK CONFIGURATION STRENGTH TO INSIST SPECIFIC CONSISTENT PROPERTIES.

	Single switch/router	Downstream switches/routers	Global switches/routers	Global network facilities
Forwarding black hole free			[14], [21], [25], [23], [22], [26]	[24]
Forwarding loop free	Impossible	[20]	[14], [21], [25], [23], [22], [26]	[24]
Per-packet consistency	Impossible		[21][25][22]	
Link congestion free	Impossible		[14][23][26]	
Network policy violation free				[24]

Solutions in traditional networks mainly focus on forwarding black hole confusion and forwarding loop confusion [8], [7], [6], [27]. From the information provided by Table.III, forwarding black hole confusion and forwarding loop confusion are weak consistent properties. The distribution character of traditional network has limited innovation of better network update schedules that aimed to hold stronger

consistent properties. The separation of control and data plane, as well as the centralized control make it possible to coordinate switches, routers and even middleboxes in SDN. Keeping stronger properties like link congestion free and network policy violation free are possible when scheduling network update.

V. Summary

Update schedule needs to be carefully treated otherwise confusions like forwarding loop, forwarding black hole, link congestion, network policy violation, etc will come out. We analyze these confusions and solutions to them in this paper. We find that more complex confusions can be solved with the help of SDN and programable interfaces provided by these solutions to help address the network update problem.

References

[1] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, pp. 749–762, Aug. 2008.

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

[3] T. Hoff., "Latency is everywhere and it costs you sales - how to crush it." http://highscalability.com/blog/2009/7/25/latency-is-everywhere-and-it-costs-you-sales-how-to-crush-it.html, July 2009.

[4] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. Su, and L. Zhang, "Improving bgp convergence through consistency assertions," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 902–911 vol.2, 2002.

[5] A. Siddiqi and B. Nandy, "Improving network convergence time and network stability of an ospf-routed ip network," in *Proceedings of the 4th IFIP-TC6 International Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems*, NETWORKING'05, (Berlin, Heidelberg), pp. 469–485, Springer-Verlag, 2005.

[6] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-bgp: Staying connected in a connected world," in *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, NSDI'07, (Berkeley, CA, USA), pp. 25–25, USENIX Association, 2007.

[7] N. K. D. Katabi and J. Wrocklawski, "A consistency management layer for inter-domain routing." Tech. Rep. MIT-CSAIL-TR-2006-006, Jan 2006.

[8] P. Francois, M. Shand, and O. Bonaventure, "Disruption free topology reconfiguration in ospf networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 89–97, May 2007.

[9] S. Raza, Y. Zhu, and C.-N. Chuah, "Graceful network state migrations," *IEEE/ACM Trans. Netw.*, vol. 19, pp. 1097–1110, Aug. 2011.

[10] L. Vanbever, S. Vissicchio, C. Pelsser, P. Francois, and O. Bonaventure, "Seamless network-wide igp migrations," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, (New York, NY, USA), pp. 314–325, ACM, 2011.

[11] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," *SIGCOMM Comput. Commun. Rev.*, vol. 35, pp. 41–54, Oct. 2005.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[13] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 3–14, ACM, 2013.

[14] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 15–26, ACM, 2013.

[15] N. Feamster and H. Balakrishnan, "Detecting bgp configuration faults with static analysis," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, (Berkeley, CA, USA), pp. 43–56, USENIX Association, 2005.

[16] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the data plane with anteater," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, (New York, NY, USA), pp. 290–301, ACM, 2011.

[17] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, (Lombard, IL), pp. 99–111, USENIX, 2013.

[18] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," *SIGCOMM Comput. Commun. Rev.*, vol. 42, pp. 467–472, Sept. 2012.

[19] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, (Berkeley, CA, USA), pp. 24–24, USENIX Association, 2012.

[20] R. Mahajan and R. Wattenhofer, "On consistent updates in software defined networks," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, (New York, NY, USA), pp. 20:1–20:7, ACM, 2013.

[21] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, (New York, NY, USA), pp. 323–334, ACM, 2012.

[22] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, (New York, NY, USA), pp. 49–54, ACM, 2013.

[23] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "zupdate: Updating data center networks with zero loss," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 411–422, ACM, 2013.

[24] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, (Berkeley, CA, USA), pp. 533–546, USENIX Association, 2014.

[25] R. McGeer, "A safe, efficient update protocol for openflow networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 61–66, ACM, 2012.

[26] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, (New York, NY, USA), pp. 539–550, ACM, 2014.

[27] J. P. John, E. Katz-Bassett, A. Krishnamurthy, and T. Anderson, "Consensus routing: The internet as a distributed system," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, (San Francisco, California), pp. p.351–364, USENIX, April 16-18 2008.