Front. Comput. Sci.
DOI 10.1007/s11704-016-6125-y

REVIEW ARTICLE

A survey of network update in SDN

Dan LI (✉)[1], Songtao WANG[1], Konglin ZHU[1,2], Shutao XIA[1]

1 Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
2 School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

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 has been investigated for many years in traditional networks where the control plane is distributed, 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 SDN solutions to these problems. Furthermore, we summarize the network configuration strength and discuss the open issues of network update in the SDN paradigm.

Keywords software defined network, network update, forwarding loop, forwarding black hole, link congestion, network policy violation

1 Introduction

Networks are not working statically. In order to keep network in a correct and efficient state, network operators need to adjust link weights, change traffic engineering (TE) schemes, migrate virtual machines and update routing policies, etc. All these actions are known as network updates, which require careful arrangement to avoid problems like forwarding loops, forwarding black holes, link congestions and network policy violations that may occur during the updating process. Indeed, the impact of network update is not trivial. It shows that about 20% network faults come from carelessly planned network update [1]. Moreover, the network updating period is not short, and network problems may occur during this update period. For instance, it showed that 30% packets suffer from loss for more than two minutes after BGP updating [2]. Neglecting the problems during the network update process is indispensable for network operators as customers' demands become more and more critical. Delay sensitive applications such as financial trade, online shopping and searching cannot tolerate any network fault that affects the quality of service (QoS) of the network. For instance, Amazon would lose 1% of the amount of sales for every 100ms of latency[1)]; while Google reported that 20% of traffic would drop for more than 500ms of search page response time.

In traditional networks where the control plane is distributed and binding together with data plane, many research efforts are spent in solving the network update problem. A straightforward idea is to reduce the convergence time of a protocol [3,4]. However, it is difficult to design a routing protocol with fast-enough convergent time considering the huge size of the Internet. Further, the innovation of new protocols should not be limited by the convergent time restriction [5]. As a result, many recent works turn attention to how to eliminate the problems during the updating process on plan, instead of mitigating them [6–9]. Greenburg et al. argued that in

Received February 29, 2016; accepted May 4, 2016

E-mail: tolidan@tsinghua.edu.cn; wangst12@mails.tsinghua.edu.cn; klzhu@bupt.edu.cn; xiast@sz.tsinghua.edu.cn

1) T. Hoff. Latency is everywhere and it costs you sales — how to crush it. http://highscalability.com/blog/2009/7/25/latency-is-everywhere-andit-costs-you-sales-how-to-crush-it.html

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" [10]. From the perspective of network update, the updating schedule produced in a distributed way that 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.

Nowadays, more and more network operators are accepting software defined network (SDN) to manage their networks. For instance, Stanford university has deployed openflow, a representative of SDN in the campus [11]. Google, Microsoft and a growing number of enterprise operators are deploying SDN in their data center networks respectively [12,13]. Thanks to the separation of control and data plane, SDN enables new solutions to the network update problem. 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 SDN. Take openflow as an example. The control plane in openflow 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 makes 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. Catching up with the trend of SDN, there is growing interest in the community to design new solutions to the network update problem. However, SDN switches still forward packets based on their own forwarding tables, thus operators need to carefully design the update mechanisms and distribute the rules to every SDN switch.

In this paper, we summarily describe the problems caused by network update as well as the solutions in a SDN paradigm. In particular, we discuss four basic confusions occurred during the network updating process, such as forwarding black hole, forwarding loop, link congestion and network policy violation. We then study the solutions to the confusions and discuss the constraints for network updating scheduling. We also carry out the discussion about the difficulties for solving different confusions. Finally, we propose the future open issues for network updating in the paradigm of SDN. 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 2 discusses the problems caused by network updating. Section 3 presents the current solution to SDN update in the literature. Section 4 discusses more issues in this problem and proposes open challenges in the network update. Section 5 concludes the paper.

2 Problems of network update

Network update problem refers to network confusions caused by careless updating schemes. Network update we mention in this paper is not a mechanism that corrects network behavior, but the change of network state managed by network operators. Actually, there have been a bunch of solutions for network static state verification [14–17], but we focus on the confusions caused by network updates. There are a list of network confusions that may occur during different network update scenarios shown in Table 1. It appears that forwarding black hole, forwarding loop, link congestion, network policy violation, etc. are in the scenarios such as updating forwarding policy, changing access list of a firewall, VM migration in Data Center and adjusting traffic engineering scheme. According to the table, it shows that majority of network update problems are caused by four basic confusions: forwarding black hole, forwarding loop, link congestion, and network policy violation. We will explain each of them in the following of this section.

Table 1 Some scenarios of network update and corresponding confusions

Scenario	Confusion
Update forwarding policy	forwarding loop, forwarding black hole, link congestion, network policy violation
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

• Forwarding black hole

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 arriving 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.

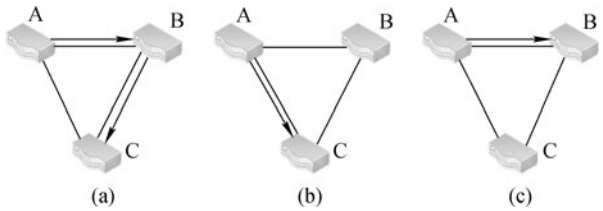


Fig. 1 An example of forwarding black hole confusion during network update

• 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 nor B can 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 the latency between control plane and forwarding plane varies with respect to different switches, any updating sequence may be possible if the control plane distributes update schemes to A, B and C at the same time. For the same reason, it is impossible to update all the switches in one instance.

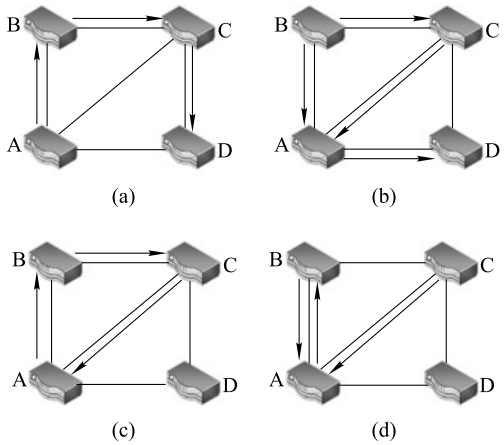


Fig. 2 An example of forwarding loop confusion during network update

• Link congestion

The link congestion refers to the situation that link is congested by flows during the network update process. An example of the link congestion confusion is shown in Fig. 3. Links are bidirectional and there are three flows in the network. 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. On the contrary, Fig. 3(d) is a well planned update scheme which puts Flow3 as the first update flow. In this case, no congestion will take place.

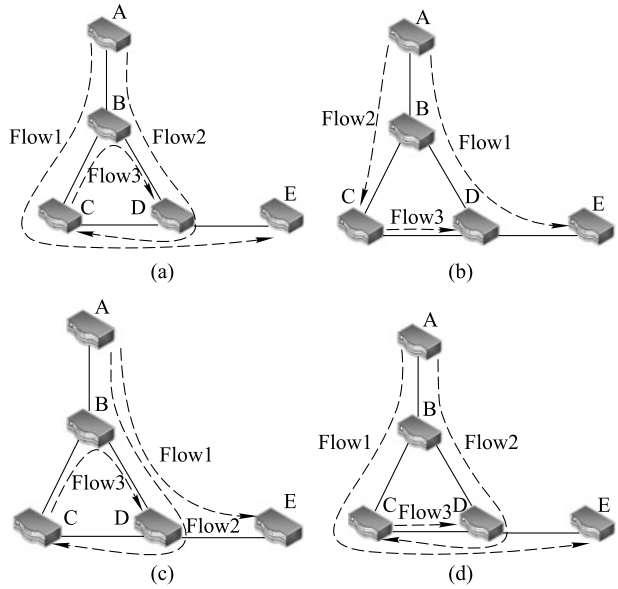


Fig. 3 An example of link congestion confusion during network update

• Network policy violation

Middleboxes (e.g., Firewall, NAT, and Proxy) are widely deployed in modern networks [18]. Packets have to follow network policies which enforce packets going through a list of middleboxes. During the network policy update, packets may violate the policies. Such confusion is known as network policy violation. We illustrate this confusion with the help of an example in Fig. 4. In the figure, all the traffic should go through a network address translator (NAT) which converts private IP address to public IP address in the packet header. Meanwhile, the traffic with 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 192.168.0.3

to 234.234.234.4 and 192.168.0.4 to 234.234.234.5, respectively (the original map is 192.168.0.3 to 234.234.234.3 and 192.168.0.4 to 234.234.234.4, as shown in Fig. 4). Thus traffics from Host1 and Host2 both go through the path of "SW1-NAT-SW1-SW2-FW-Internet". As a result, Host1 contacts the Internet now, which violates the network policy that Host1 cannot contact the Internet; meanwhile, traffic from Host2 also have to go through FW, which increases the load of FW.

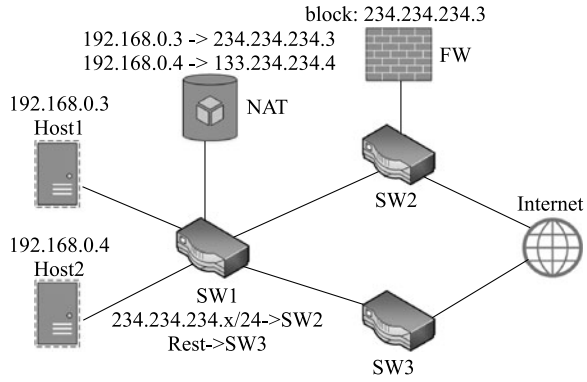


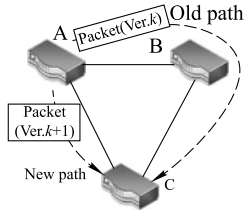
Fig. 4 An example of network policy violation confusion during network update

3 SDN solutions to network update

In this section, we firstly illustrate solutions to the four basic confusions discussed above. Then we will discuss three limitations that will affect performance of network update scheduling.

3.1 Solution to forwarding black hole

The forwarding black hole results from the reason that there is no forwarding rule matching the packet. Mahajan and Wattenhofer propose a solution to this confusion as "add before remove" in Ref. [19]. The scheme suggests that switches should update new rules first before removing old rules. Then Reitblatt et al. [20] implement a more generous version of "add before remove" on the whole network to solve the forwarding black hole confusion. We illustrate the solution of [20] by an example shown in Fig. 5. In the example, 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, while 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 with version k. 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

Indeed, 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 goes 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 only keep forwarding black hole free.

3.2 Solution to forwarding loop

In order to solve forwarding loop confusion, we need to avoid the existence of the loop during the update process. We have mentioned that "per-packet consistency" by Reitblatt's solution [20] is a strong consistent property in last subsection. Indeed, such "per-packet consistency" not only keeps the forwarding black hole free but also holds forwarding loop free. 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. A packet is forwarded along either the path shown in Fig. 2(a) or the path shown in Fig. 2(b) but never a mix of the them.

However, Mahajan and Wattenhofer point out that Reitblatt's solution is too strong for forwarding loop free property [19]. 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 Mahajan and Wattenhofer's solution to forwarding loop confusion. Fig. 6(a)

shows the forwarding path to node D before network updating and Fig. 6(d) shows the forwarding path to node D after network updating. All subfigures (a)–(d) in Fig. 6 in sequence indicate the update schedule according to Mahajan and Wattenhofer's solution. The update sequence is A-C-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 the root of the dependency tree; node A is the child of node D and node B and node C are children of node A, resulting from node D is the next hop for node A and node A is the next hop for node B and C, as shown in Fig. 6(d). After building up the dependency tree, the updating rule will be scheduled as updating the root first, and then update children of each node in sequence following the seemingly breadth first search (BFS) rule.

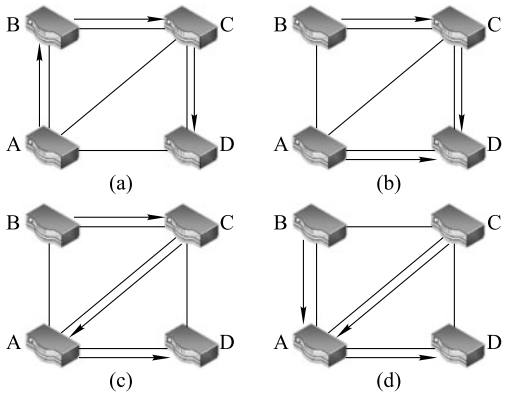


Fig. 6 An example solution to forwarding loop confusion during network update

3.3 Solution to link congestion

The link congestion confusion is caused by the overloading of link during the update process. Microsoft demonstrates their inter-data center network solution to link congestion SWAN [13] in 2013. The main purpose of SWAN is to highly utilize the network capacity even the traffic volume varies significantly by time. SWAN leaves "scratch capacity" on links. "Scratch capacity" will not be used until updating, and it 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 ICU [21]. They both trade updating time with scarce resource which in SWAN is link capacity and in ICU [21] is TCAM. zUpdate [22] by Liu et al. aims to eliminate congestion in data center updating progress. They point out that in data centers switches utilize equal cost multipath routing (ECMP) to split traffic evenly among all next hops to make fully use of the redundant paths [22]. If one link fails, traffic on this link is evenly split to the remaining links which may cause congestion. zUpdate breaks the fairness among multiple paths and split effected traffic on each path carefully when failure happens. Like [20], optimization programming model in zUpdate only requires operator to provide final configuration without paying attention to update details.

3.4 Solution to network policy violation

The origin of network policy violation is that a packet cannot be matched with its original address. Fayazbakhsh et al. point out that the key to solving network policy violation confusion is "OriginBinding" [23]. 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. Also, They compare tags with the mapping received from the controller and find out the original IP of packets, so that 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. In such a manner, network policy always holds for every packet.

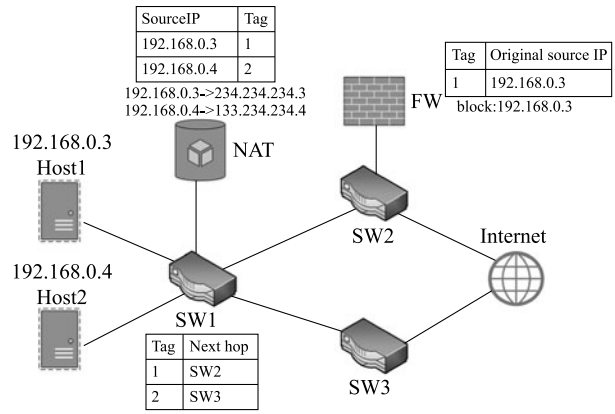


Fig. 7 An example solution to network policy violation confusion during network update

Fayazbakhsh's solution helps switches to "OriginBind" a packet with its original IP address, but there is still a problem that may violet security polity. In a case which is similar to Fig. 7, Host1 is multihoming to SW3. Therefore, flows from Host1 may visit Internet through SW3. How can we let flows from Host1 go back to SW2 and blocked by the FW? Solution to the problem is called waypoint enforcement provided by Ludwig et al. [24]. In this case, SW2 is the waypoint that

every flow from Host1 must go through. Ludwig's solution could guarantee that during update or any cases, flows will always travel through the selected way point and therefore security policy is hold.

3.5 Constraints of network update scheduling

Besides the solutions to basic confusions, there are several studies investigating the constraints of network update scheduling. In this paper, we mainly focus on three types of constraints: memory constraint, update efficiency and customizable consistency properties.

1) Memory constraint As TCAM is the most crucial memory resource for switches, too much TCAM consumption caused by updating mechanism may result in the inefficiency of network management. Reitblatt's solution is an effective but inefficient updating mechanism which consumes 50% of TCAM capacity more. Therefore, some researches focus on releasing the over consumed memory. McGeer [25] tries to use SDN controller as a cache during update process. As shown in Fig. 8, packets affected by the update process are all sent to the controller. After update process, these packets are sent back to network for further forwarding. The solution consumes no additional memory but leaves 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 (e.g., Openflow channel). This is a disaster for network management for the reason that fraction of configuration command may get lost. McGeer admits that the solution can only be deployed in very special cases. Katta's solution [21] 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 Zipf's law (an exponentially decaying distribution), then "80% of traffic could be updated in the first round and 99% of traffic could be updated after three rounds which is acceptable in real network environment [21]."

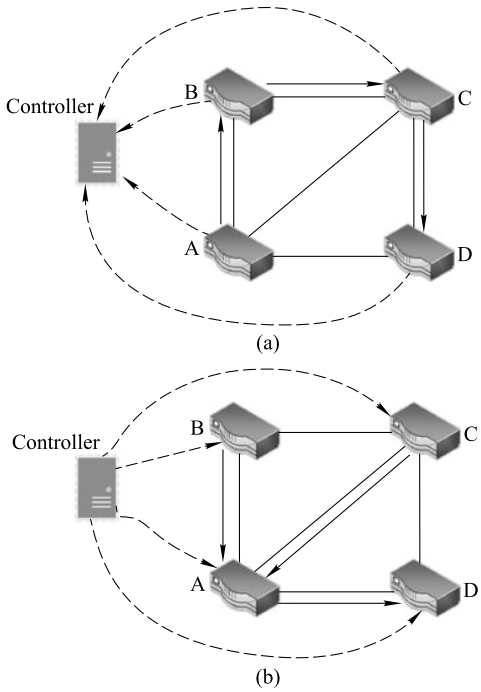


Fig. 8 Mechanism of McGeer's network update schedule

2) Update efficiency Update efficiency is critical, as the careless and slow update may result in the chaos of network behaviors. Jin et al. find that the network updating efficiency is quite low [26]. They find a way to speed up this process of keeping link congestion free. We use the updating example as shown in Fig. 9 to explain the scheme. Figure 10 is the dependency graph of this updating scheme, which illustrates how the scheme updates the network. Rectangle represents available resource on nodes and links. For example "SD:50" means 50 units of space are available in node D, and "SC-SD:5" means 5 units of space are available on link between node C and node D. Circle represents operation. For example, circle A means add rule for Flow4 at node C. The detailed illustration regarding operations is listed in Table 2. Triangle represents the flaws on which the operations work. 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 lists all resources remained and transformation of resources through operations. Besides, Jin et al. develop an algorithm to find a tree through all operations which obey resource limit. Resulting sequence of operations along the tree is the solution to update network efficiently. The approaches which are classified as "update ordering" (another class is two phase update) are all "trying" to find an order of rule update, but such an order does not always exist. Foerster et al. in their work [27] prove that for splittable flows, whether congestion free update order exists can be verified in polynomial time. If such an order does not exist, those "update ordering" approaches may stuck in calculating the order.

Paris et al. [28] try to solve the fundamental problem of online SDN controllers to decide when to perform flow reconfigurations for efficiently network updating. They formulate a stochastic optimization problem to minimize the actual network cost by selecting the reconfiguration instances sub-

ject to a time-average constraint on the frequency of network reconfigurations, so that no more than one network reconfigurations occur per iteration of the solver.

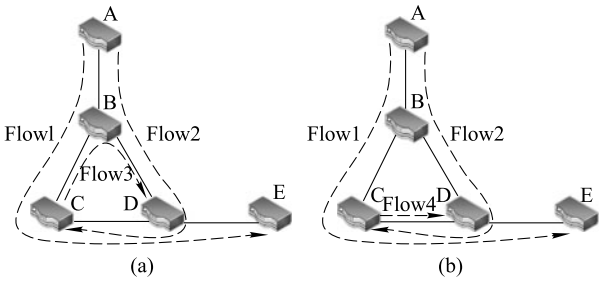


Fig. 9 A scenario of network update efficiency

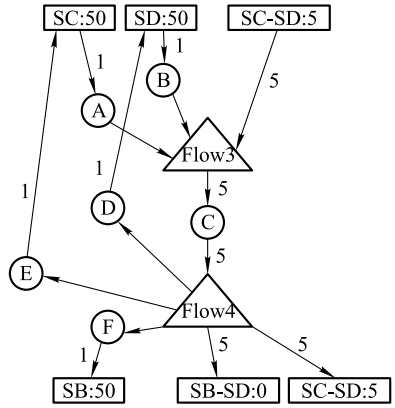


Fig. 10 Dependency graph of Fig. 9

Table 2 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

3) Customizable consistency properties Network operators desire a network updating mechanism that can satisfy with customizable consistency properties. The network update is to ensure customizable correctness properties as the network evolves efficiently. However, due to the distributed nature of networks, it may result in the uncertainty of network that the network behavior deviates from desired properties as the instilling of updates. Zhou et al. propose a general algorithm for various consistency properties [29]. It introduces customizable consistency generator (CCG) to achieve the customizable network updates. In particular, the CCG collapses all possible states onto one forwarding graph and orders the updates into a queue and checks the safety of the queued updates. A verification engine traverses on the forwarding graph to ensure the correctness of the instilling updates. Then it confirms the updates with network model.

4 Discussion

Mahajan and Wattenhofer have revealed the relationship between the strength of consistent properties and dependency structure in their work [19]. Their conclusion is that stronger consistent properties require broader dependency structures to hold consistency. We borrow the idea of Ref. [19] and draw Table 3 filled with consistent properties and solutions that are introduced in this paper. The first column of Table 3 lists consistent properties in the order of strength and the first row of Table 3 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 Ref. [19]. 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.

Solutions in traditional networks mainly focus on forwarding black hole confusion and forwarding loop confusion [5–7,30]. From the information provided by Table 3, forwarding black hole confusion and forwarding loop confusion are weak consistent properties. The distribution character of traditional network has limited innovation of better network

Table 3 Network configuration strength to insist specific consistent properties

	Single switch/router	Downstream switches/routers	Global switches/routers	Global network facilities
Forwarding black hole free			[13,20–22,25,26]	[23]
Forwarding loop free	Impossible	[19]	[13,20–22,25,26]	[23]
Per-packet consistency	Impossible		[20,21,25]	
Link congestion free		Impossible	[13,22,26]	
Network policy violation free				[23]

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

All the works we present by far have a consensus that atomic update does not exist because switches cannot update at exactly the same time. Updating duration time and controller response time vary a lot and are not predictable. Therefore, update ordering and two phase updating are two main research directions on network update problem. However, Mizrahi et al. in their work [31] show that accurate time based update is possible. This is an exciting progress and may make network update faster than ever.

Although researchers have paid a lot of efforts on network updating problem, there are still open challenges need to be addressed. Firstly, the state-of-the-arts for solving confusions during network updates treat every flow equally. In reality, some flows are more emergent than the others due to its upper layer service. For example, the flows for live streaming are more latency sensitive than the flows for email service. Update strategy should consider these emergent flows first before those who are not sensitive to latency. Because SDN can schedule each flow accurately, we think priority based flow update mechanism will be a direction in the future research. Secondly, papers aimed at congestion free consistency usually assume that link will not suffer from congestion. Such assumption is hardly to hold in a busy network. Network update method should be capable of reconfiguring under link congestions meanwhile satisfying some consistencies (e.g., the network can update in the case of congestion, and congestion should not exceed 10 percents of link capacity during the update). Thirdly, as the scale of SDN keeps increasing, multiple controllers proposal is suggested [32]. Scalability is nontrivial for future development of SDN. SDN enables the capability of centralized network configuration that allows the network operators to have more opportunities for solving the above-mentioned network update issues.

5 Summary

Update schedule needs to be carefully treated, otherwise confusions like forwarding loop, forwarding black hole, link congestion, network policy violation 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.

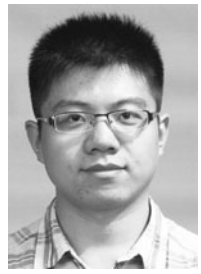
Acknowledgements The work was supported by the National Key Basic Research Program of China (973 program) (2014CB347800), the National Natural Science Foundation of China (Grant Nos. 61522205, 61432002, 61133006, and 61502045), the National High-tech R&D Program of China (863 program) (2013AA013303, 2015AA01A705, and 2015AA016102), EU FP7 Marie Curie Actions project Grant Agreement (the Cleansky project) (607584), ZTE corporation and Tsinghua University Initiative Scientific Research Program.

References

1. Markopoulou A, Iannaccone G, Bhattacharyya S, Chuah C N, Ganjali Y, Diot C. Characterization of failures in an operational IP backbone network. IEEE/ACM Transactions on Networking, 2008, 16(4): 749–762
2. Labovitz C, Ahuja A, Bose A, Jahanian F. Delayed Internet routing convergence. IEEE/ACM Transactions on Networking, 2001, 9(3): 293–306
3. Pei D, Zhao X L, Wang L, Massey D, Mankin A, Su S F, Zhang L X. Improving BGP convergence through consistency assertions. In: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies. 2002, 902-911
4. Siddiqi A, Nandy B. Improving network convergence time and network stability of an OSPF-routed IP network. In: Proceedings of International Conference on Research in Networking. 2005, 469–485
5. Kushman N, Kandula S, Katabi D, Maggs B M. R-BGP: staying connected in a connected world. In: Proceedings of Symposium on Networked Systems Design and Implementation. 2007
6. Kushman N, Katabi D, Wroclawski J. A Consistency Management Layer for Inter-Domain Routing. Technical Report. 2006
7. Francois P, Shand M, Bonaventure O. Disruption free topology reconfiguration in OSPF networks. In: Proceedings of the 26th IEEE International Conference on Computer Communications. 2007, 89–97
8. Raza S, Zhu Y, Chuah C N. Graceful network state migrations. IEEE/ACM Transactions on Networking, 2011, 19(4): 1097–1110
9. Vanbever L, Vissicchio S, Pelsser C, Francois P, Bonaventure O. Seamless network-wide IGP migrations. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 314–325
10. Greenberg A, Hjalmtysson G, Maltz D A, Myers A, Rexford J, Xie G, Yan H, Zhan J B, Zhang H. A clean slate 4D approach to network control and management. ACM SIGCOMM Computer Communication Review, 2005, 35(5): 41–54
11. Mckeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2010, 38(2): 69–74
12. Jain S, Kumar A, Mandal S, Ong J, Poutievski L, Singh A, Venkata S, Wanderer J, Zhou J L, Zhu M, Zolla J, Hölzle U, Stuart S, Vahdat A. B4: experience with a globally-deployed software defined WAN. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 3–14
13. Hong C Y, Kandula S, Mahajan R, Zhang M, Gill V, Nanduri M, Wattenhofer R. Achieving high utilization with software-driven WAN. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 15–26

14. Feamster N, Balakrishnan H. Detecting BGP configuration faults with static analysis. In: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, 2015, 43–56
15. Mai H, Khurshid A, Agarwal R, Caesar M, Godfrey P, King S T. Debugging the data plane with anteater. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 290–301
16. Kazemian P, Chang M, Zeng H, Varghese G, McKeown N, Whyte S. Real time network policy checking using header space analysis. In: Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation. 2013, 99–112
17. Khurshid A, Zhou W, Caesar M, Caesar M, Godfrey P B. VeriFlow: verifying network-wide invariants in real time. ACM SIGCOMM Computer Communication Review, 2015, 42(4): 467–472
18. Sekar V, Egi N, Ratnasamy S, Reiter M K, Shi G. Design and implementation of a consolidated middlebox architecture. In: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation. 2012, 323–336
19. Mahajan R, Wattenhofer R. On consistent updates in software defined networks. In: Proceeding of the 12th ACM Workshop on Hot Topics in Networks. 2013, 29–31
20. Reitblatt M, Foster N, Rexford J, Schlesinger C, Walker D. Abstractions for network update. ACM SIGCOMM Computer Communication Review, 2015, 42(4): 323–334
21. Katta N P, Rexford J, Walker D. Incremental consistent updates. In: Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined NETWORKING. 2013, 49–54
22. Liu H H, Wu X, Zhang M, Yuan L, Wattenhofer R, Maltz D. zUpdate: updating data center networks with zero loss. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 411–422
23. Fayazbakhsh S K, Chiang L, Sekar V, Yu M, Mogul J C. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In: Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation. 2014, 543–546
24. Ludwig A, Rost M, Foucard D, Schmid S. Good network updates for bad packets: waypoint enforcement beyond destination-based routing policies. In: Proceedings of the 13th ACM Workshop on Hot Topics in Networks. 2014
25. Mcgeer R. A safe, efficient update protocol for OpenFlow networks. In: Proceedings of the 1st ACM Workshop on Hot Topics in Software Defined Networks. 2012, 61–66
26. Jin X, Liu H H, Gandhi R, Kandula S, Mahajan R, Zhang M, Rexford J, Wattenhofer R. Dynamic scheduling of network updates. ACM SIGCOMM Computer Communication Review. 2014, 44(4): 539–550
27. Brandt S, Förster K T, Wattenhofer R. On consistent migration of flows in SDNs. In: Proceedings of IEEE INFOCOM. 2016
28. Paris S, Destounis A, Maggi L, Paschos G, Leguay J. Controlling flow reconfigurations in SDN. In: Proceedings of IEEE INFOCOM. 2016
29. Zhou W, Jin D, Croft J, Caesar M, Godfrey P B. Enforcing customizable consistency properties in software-defined networks. In: Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation. 2015, 73–85
30. John J P, Katz-Bassett E, Krishnamurthy A, Anderson T, Venkataramani A. Consensus routing: the Internet as a distributed system. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation. 2008, 351–364
31. Mizrahi T, Rottenstreich O, Moses Y. TimeFlip: scheduling network updates with timestamp-based TCAM ranges. In: Proceeding of the 2015 IEEE Conference on Computer Communications (INFOCOM). 2015, 2551–2559
32. Guo Z H, Su M, Xu Y, Duan Z M, Wang L, Hui S F, Chao H J. Improving the performance of load balancing in software-defined networks through load variance-based synchronization. Computer Networks, 2014, 68(11): 95–109

Dan Li is an associate professor in Computer Science Department of Tsinghua University, China. He received his PhD degree in computer science from Tsinghua University in 2007. His research interests include future Internet architecture and data center networking. He is an awardee of the NSFC Excellent Young Scholars Program in 2015.



Songtao Wang received his master degree in system on chip from Southampton University, UK in 2011. Now he is a PhD student in Tsinghua University, China. His research interests include network update and software defined datacenter network.

Konglin Zhu received his master degree in computer science from University of California, Los Angeles, USA in 2009 and his PhD degree in University of Goettingen, Germany in 2014 respectively. He is now an assistant professor at Beijing University of Posts and Telecommunications, China. He is also a visiting scholar at Tsinghua University, China sponored by FP7 CleanSky project. His research interests include network virtualization and vehicular networks.

Shutao Xia received the BS degree in mathematics and the PhD degree in applied mathematics from Nankai University, China in 1992 and 1997, respectively. Since January 2004, he has been with the Graduate School of Shenzhen of Tsinghua University, China, where he is currently a professor. His current research interests include coding theory, information theory and networking.