VLAN-PSSR: PORT-SWITCHING BASED SOURCE ROUTING USING VLAN TAGS IN SDN DATA CENTERS

Ovidiu Mihai PONCEA[1], Florica MOLDOVEANU[2], Victor ASAVEI[3]

Source Routing allows a node in the path of a packet to specify partially or completely the route taken by that packet. The route is appended to packet header as a list of nodes to traverse, therefore making simple, stateless forwarding decisions. In this paper, we present a novel approach of Source Routing in SDN that uses stacked VLAN tags. Our solution was validated in Mininet using the Ryu controller and proven to have multiple advantages over other forwarding methods.

Keywords: Networking, SDN, Source Routing, Mininet, VLAN

1. Introduction

In Source Routing, routes are usually specified when packets enter a network either at *source* or at one of the *edge* switches. Many custom implementation of source routing exists (mainly in HPC – High Performance Computing solutions) such as Myrinet [1], Quadrics [2] and IEEE 1355.

In standard computer networking, IP provides a special header, *Loose Source Routing*, which can be used to specify a list of routers that a packet can take. At each node, packet destinations are replaced with information from this list so that a packet can tunnel through a network that otherwise is unable to forward it. This is intended to provide mobility for users through multiple provider networks. To note here that this option can become a security hazard as it may be used to piggyback packets to destinations that would otherwise be unreachable. The solution is limited to IP only and most internet routers disable it.

In SDN, the controller keeps the global view of the network, it controls the forwarding nodes, and knows what hosts are connected to edge switches. Therefore, it is much easier for it to build and set routes at the edge of the network for all the packets entering it. Furthermore, OpenFlow (www.goo.gl/yC79ge) has the necessary mechanisms to create this kind of behavior without modifications.

[1] PhD student, Dept of Computer Science and Engineering, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: ovidiu.poncea@cs.pub.ro

[2] Prof, Dept of Computer Science and Engineering, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: florica.moldoveanu@cs.pub.ro

[3] Lect., Dept of Computer Science and Engineering, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: victor.asavei@cs.pub.ro

In Data Center computer networking in general, and SDN in particular, there are three main techniques for packet forwarding:

1. Destination based forwarding – packets destinations (e.g. MAC or IP destination addresses) are matched against a list of destinations and, when an entry matching the searched address is found, packet is forwarded on the correct port using forwarding information from the matched list entry.
2. Label based forwarding – packets entering the network are classified and a label is appended to each packet, then forwarded based only on that label (e.g. ATM and MPLS).
3. Source routed based forwarding – packets are forwarded based on a list of nodes specified in the packet itself.

A list of different forwarding techniques is presented in [4]. The paper analyzes CONGA [5], Shadow MACs [6], XPath [7], FastPass [8], SlickFlow [9] and SecondNet [10].

The question is, why source routing? SDN can already forward a packet through the network so, what benefits can source routing bring? The following sections will explore this and present a novel source routing solution that provides both unicast and multicast. The proposed source routing solution is based on stacked VLAN tags that are pushed at edge and popped at each node after forwarding is decided. A similar solution using MPLS tags is presented in [3]. The main advantage of VLAN stacking over MPLS is that support for MPLS is limited in core switches while VLAN is much more common. Some switches do not have MPLS support while the majority only support 3 levels of tags. Even if this number can be increased it is recommended only for networks with small diameter. Also, header sizes are smaller with VLAN tags – 2 bytes versus 4 bytes for MPLS (+2 bytes for header type in both cases).

VLAN based source routing only needs VLAN forwarding and popping – a common feature in current generation OpenFlow hardware switches. Even though these switches are usually unable to *push* more than 4 tags, they are able to easily *pop* a single tag and forward based on it while keeping the rest of tags intact. Pushing many tags is only required by edge switches which usually are software switches, therefore easier to implement[4].

2. Limitations of destination/labels based routing and advantages of Source Routing

Source routing reduces the following limitations of SDN standard destination based forwarding or tag based forwarding:

[4] Open vSwitch support for multiple VLAN tags is under review and will be available in the next official release.

1. Limitations of flow tables – with source routing table usage drops dramatically [4]. We will show bellow that our method needs a static number of flows in core switches. Number of flows increases in edge switches but, since they are software, this is not a major issue.
2. Slow network updates – network updates can be slow when many flows need to be updated at once. The slowness come both from the controller and from the switches themselves. Source routing can reduce this issue as the number of flows in core switches is reduced. Our approach does not involve any update of flow tables in switches other than ones from the edge.
3. Traffic engineering can be complex and multipath routing hard to implement. With source routing flows can easily be scheduled from edge to go on multiple paths and packet distribution can be better controlled. In fact, the edge can choose a different route on a per-packet basis.

Limitations of source routing:

1. Failover can be hard to implement, once a device on the path of a source routed packet fails, the switches neighboring the failed device no longer know how to forward those packets; a solution would be to just send them back to the controller but this may overwhelm it.
2. Source routing based on switch ID's may be more resilient to failover than port based ones because, if a port fails, packets may be forwarded to a neighbor that is aware of the next ID in the path and may reroute around the failed device so that packets returns to the previous hop in the route.

Full broadcasts and multicast is not supported with source routing as multiple destinations are almost impossible to specify. Our solution is partial yet usable as it covers most use cases in real world Data Centers.

3. Description of the VLAN-PSSR solution

In the Ethernet header, VLAN tags sit between source MAC address and higher protocols headers, usually IPv4 or IPv6. Multiple tags are appended to the packet one after the other. In the Ethernet header, these tags are identified by an Ethertype of 0x8100, therefore each tag contains these two bytes. Only after Ethertype we have VLAN specific information:

- Priority code point (PCP), a 3 bit-field which maps a packet to a priority queue,
- Drop Eligible Indicator (DEI) – single bit that indicates if packets are eligible for dropping in case of congestions and
- VLAN ID (VID) – a 12 bit field specifying the VLAN to which the packet belongs.

Our solution, VLAN-PSSR reuses VID for source routing. One bit is used for specifying if the tag is multicast or unicast. For unicast 8 bits specify the port number of a switch (0 to 255) while 3 bits are not used (Fig. 1).

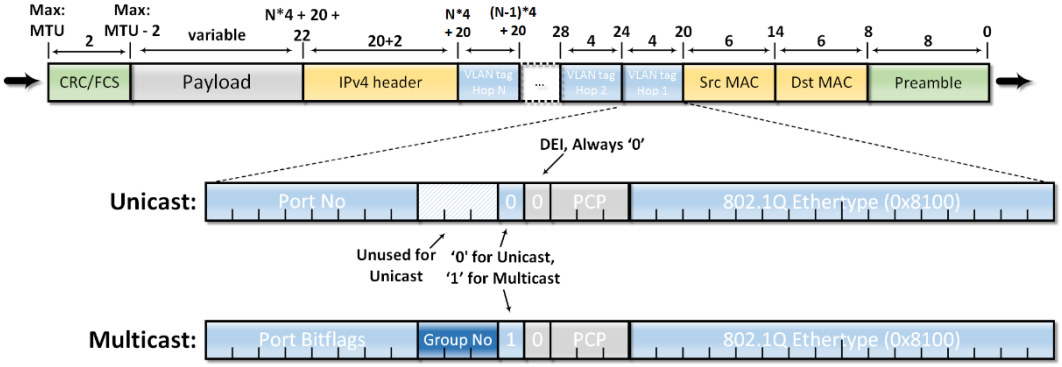

Fig. 1. VLAN-PSSR Packet Format

For multicast, we divide the 11 bits of VID in two parts:

1. Group *number* – an unsigned integer pointing to a subset of ports. If we divide the total number of ports of a switch in groups, this number represents one such division.
2. Port *bitflags* – a set of bits representing all ports of a group. Each bit corresponds to a single port. Multiple ports can be selected at the same time. Therefore, a packet is forwarded on all ports that are part of that group and have their bits set to 1 in port bitflags

For example, in Fig. 1, 3 bits are reserved for *group number* and 8 for *port bitflags*. In this configuration, we can define 8 groups, each with 8 ports, for a total of 64 ports.

Note that we can remove the multicast bit and consider any message that has a group number higher than '0' to be multicast. In this case, we can multiplex (duplicate) a packet up to 120 ports. If we need more ports, we can go further and increase the *group bitflag* to 5 bits and decrease the port bitflags to 7 bits resulting in 217 ports. A higher multiplexing also increases the number of tags needed which increases packet size. Therefore, the optimal number of tags should be selected based on the number of ports that a switch has.

The maximum number of ports (P) and max groups (G) can be computed with:

$$\begin{cases} P = (2^{Gs} - 1) * Ppg \\ G = 2^{Gs} \\ Gs + Ppg = 11 \end{cases} \quad (1)$$

Where *Gs* is the number of bits reserved for group number, *Ppg* is the number of bits in *Port bitflag* (i.e. ports per group) and 11 is the number of bits in VID.

4. VLAN-PSSR functional validation for unicast

To validate our solution, we first implemented it in Mininet (www.mininet.org) with a small configuration (Fig. 2) and verified that ping is successful and that UDP and TCP data connections can be successfully established. We then validated the message content with Wireshark (www.wireshark.org). To make the setup work we used the latest version of Open vSwitch (www.openvswitch.org) from the development branch (v 2.5) and applied a patch for allowing multiple VLAN tags[5]. Open vSwitch was then connected to Ryu SDN controller (https://osrg.github.io/ryu/) and on top of Ryu we implemented our VLAN-PSSR application.

Mininet setup consists of three hosts H_1, H_2 and H_3 and 5 switches S_1, S_2 and S_3 at the edge and C_{11} and C_{12} at core. Hosts are Linux containers instances[6] and switches are Open vSwitch instances (bridges) connected to Ryu controller. Flows are then managed by our VLAN-PSSR implemented on top of Ryu.

We configured Ethernet MAC addresses equal to host number (for easier identification) and OpenFlow datapath IDs (dpid) to the switch number (1, 2, 3, 11=0xb & 12=0xc). Communication between H_2 and H_3 uses destination based forwarding, without source routing, as they are only one hop away while communication between H1 ⇔ H2 and H1⇔ H3 uses Source Routing. The validation configuration is presented in Fig. 2.

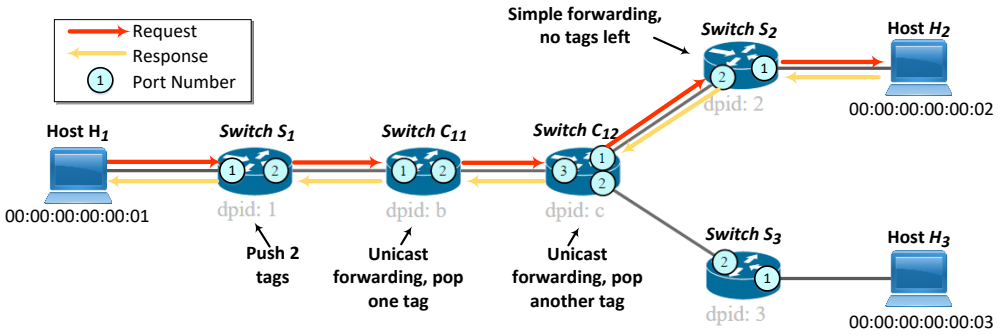

Fig. 2. VLAN-PSSR unicast validation setup

PSSR tags are added in the edge switches (S_1, S_2 and S_3) and removed (popped) in core at each switch. As an example, in Fig 2, we have a request and

[5] This patch is currently under review and will be available in next official Open vSwitch release

[6] This is an *operating system level virtualization* solution provided by the Linux kernel. It offers logical isolation of process, networking and file system resources between containers so that any one container is unable to access resources from other containers.

response path between source H_1 and destination H_2 where tags are pushed in S_1, with the flows in Table 1, and forwarded between core switches, with flows in Table 2. Flows in blue are used for management and the other ones implement VLAN-PSSR. Management flows forward packets to controller if no match is found, flood ARP requests and allow LLDP neighbor discovery.

Table 1 shows the process of matching a packet header and pushing all needed VLAN-PSSR tags. When a packet enters the switch it is matched against its destination (e.g. request in Fig. 2 has a destination address of 00:00:00:00:00:02) and a *tag* is added, then it is sent to the next table in the pipeline, matched again and *another tag* added. In the end it is forwarded to next hop (C_{11}) that is connected to port 2:

- flow #3: destination 00:00:00:00:00:02 is matched, first tag is added (for crossing $C_{11} \rightarrow C_{12}$) & sent to table 1
- flow #7: destination matched again, second tag is added (for crossing $C_{12} \rightarrow S_2$) & packet is sent to port 2.

Table 1

Unicast flow table of switch S_1

No.	Flow entries
1.	table=0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
2.	table=0, priority=60000,dl_dst=ff:ff:ff:ff:ff:ff actions=FLOOD
3.	table=0, in_port=1,dl_dst=00:00:00:00:00:02 actions=push_vlan:0x8100,set_field:4097->vlan_vid,goto_table:1
4.	table=0, in_port=1,dl_dst=00:00:00:00:00:03 actions=push_vlan:0x8100,set_field:4098->vlan_vid,goto_table:1
5.	table=0, in_port=2,dl_dst=00:00:00:00:00:01 actions=output:1
6.	table=0, priority=0 actions=CONTROLLER:65535
7.	table=1, in_port=1,dl_dst=00:00:00:00:00:02 actions=push_vlan:0x8100,set_field:4098->vlan_vid,output:2
8.	table=1, in_port=1,dl_dst=00:00:00:00:00:03 actions=push_vlan:0x8100,set_field:4098->vlan_vid,output:2

At core switches, in Table 2, packets are matched against their VLAN tags and forwarded to corresponding ports (e.g. match on VID 1 will forward to Port 1); see flows #3 to #17. Before forwarding packets to their outputs a tag is *poped* from the stacked list of VLANs.

Table 2

Unicast flow table of switch C_{11} & C_{12}

No.	Flow entries
1.	table=0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
2.	table=0, priority=60000,dl_dst=ff:ff:ff:ff:ff:ff actions=FLOOD
3.	table=0, dl_vlan=1 actions=pop_vlan,output:1
4.	table=0, dl_vlan=2 actions=pop_vlan,output:2
5.	table=0, dl_vlan=3 actions=pop_vlan,output:3
...	[cut 12 entries]
17.	table=0, dl_vlan=15 actions=pop_vlan,output:15
18.	table=0, priority=0 actions=CONTROLLER:65535

Looking at the tables in core switches we see that they are static in size and depend linearly on the number of ports, so for a 64 port switch only 64 static entries are needed. With destination or label based forwarding managing

thousands of flows in each core switch is normal but, with source routing, we can substantially reduce this number to a maximum of a few hundred.

5. VLAN-PSSR functional validation for multicast

For VLAN-PSSR multicast packets are transmitted on a unicast path until penultimate hop where packets are multiplied and sent to the last hop for final forwarding. The reason for doing this is that VLAN-PSSR can only do a single multiplication and this needs to be close to the packet destination. In data centers usually the penultimate hop is the Top of Rack (ToR) switch while last hop is the virtual switch of servers. Therefore, our solution is providing multicast inside a single rack but, since we are targeting multitenant Data Centers with edge virtual switches, multicasting inside the same rack represents the majority of use cases. Broadcast domains are usually small in multitenant Data Centers with only a few VMs connected to the same domain (around 10 - 20) which, to reduce bandwidth usage of the core network, are kept closely together, rarely spanning multiple racks.

For validating our approach, we used the same setup as before, with a multicast stream originating in H_1 and sent to both H_2 and H_3 (Fig. 3). From H_1 to C_{12} packets are transmitted using unicast and multiplication is done by C_{12}.

At edge switch S_1 both unicast and multicast tags are added, unicast first and multicast last so that, when packet arrive at C_{12}, only the multicast tags are left. Then C_{12} multiplies the packet and forwards it to both destination edge switches. To note that C_{12} is unable to drop multicast tags so the edge switches need to pop any remaining tags before sending the packet to the destination host[7].

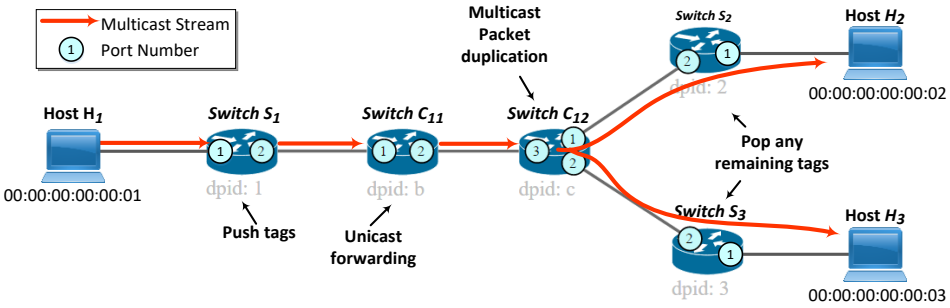

Fig. 3. VLAN-PSSR Multicast setup

[7] This is only valid for OpenFlow 1.3. Higher versions have two pipelines one on *ingress*, which processes packets when they enter the switch and another one on *egress*, which processes packets after *output* port action has been decided. Therefore, multiplication is done on *ingress* and VLAN tag drop can happen on *egress*. We used version 1.3 capabilities as these are more widespread.

Flow table of edge switch S_1 is presented in Table 3. Management flows are in blue, flows that add VLAN tags in black, flows that drop all VLAN tags remaining are in red.

Table 3

Flow Table of Edge switch S_1 in multicast case

No.	Flow entries
1.	table=0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
2.	table=0, priority=65535,vlan_tci=0x1800/0x1800 actions=pop_vlan,TABLE
3.	table=0, priority=60000,dl_dst=ff:ff:ff:ff:ff:ff actions=FLOOD
4.	table=0, priority=0 actions=CONTROLLER:65535
5.	table=0, in_port=1,dl_dst=00:00:00:00:00:02 actions=push_vlan:0x8100,set_field:4097->vlan_vid,goto_table:1
6.	table=0, in_port=1,dl_dst=00:00:00:00:00:03 actions=push_vlan:0x8100,set_field:4098->vlan_vid,goto_table:1
7.	table=0, in_port=1,dl_dst=01:00:5e:00:00:01 actions=push_vlan:0x8100,set_field:6150->vlan_vid,goto_table:1
8.	table=0, in_port=2,dl_dst=00:00:00:00:00:01 actions=output:1
9.	table=1, in_port=1,dl_dst=00:00:00:00:00:02 actions=push_vlan:0x8100,set_field:4098->vlan_vid,output:2
10.	table=1, in_port=1,dl_dst=00:00:00:00:00:03 actions=push_vlan:0x8100,set_field:4098->vlan_vid,output:2
11.	table=1, in_port=1,dl_dst=01:00:5e:00:00:01 actions=push_vlan:0x8100,set_field:4098->vlan_vid,output:2

Flow tables of nodes doing multicast forwarding (penultimate hops) are complex (Fig. 4). Processing is done in a pipeline starting at table 0 and each pass processes a single multicast tag, therefore if multiple tags are present multiple passes of the same packet through the pipeline are needed, which decreases performance when used in software.

Processing takes the following steps (P is the number of ports in a group and G the number of groups):

1. **In table #0**, multicast tag is identified. If the packet is multicast tagged processing continues in table 10;
2. **In table #10** packet is matched against a single entry and sent to a port, otherwise it is sent to next table;
3. **In table #11 to #10 + (P-1)** packet is matched against other group/port pair until reaching end of pipeline;
4. **In table #10 + P,** if packet still contains multicast tags it is sent back to the beginning of pipeline to process another tag otherwise it is considered processed and dropped.

Edge switch tables (Table 4) only pop VLAN and provide destination based forwarding to H1; for our experiments we used multicast Ethernet group 01:00:5e:00:00:01. With blue we marked management flows, with red unicast flows and with black are the two multicast flows.

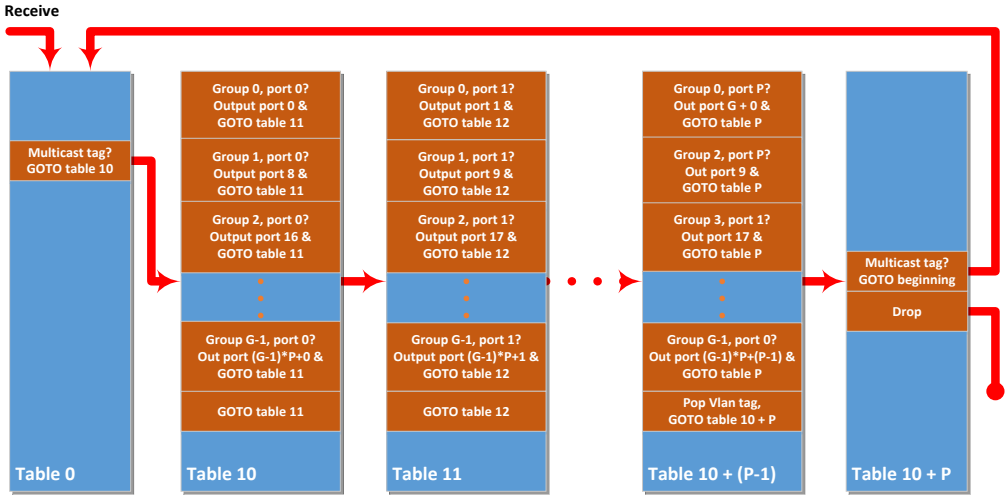

Fig. 4. Multicast flow entries of core switches

Table 4

Flow Table of Edge switch S_2 in multicast case

No.	Flow entries
1.	table=0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
2.	table=0, priority=60000,dl_dst=ff:ff:ff:ff:ff:ff actions=FLOOD
3.	table=0, priority=0 actions=CONTROLLER:65535
4.	table=0, priority=65535,vlan_tci=0x1800/0x1800 actions=pop_vlan,TABLE
5.	table=0, dl_dst=01:00:5e:00:00:01 actions=output:1
6.	table=0, in_port=1,dl_dst=00:00:00:00:00:03 actions=push_vlan:0x8100,set_field:4098->vlan_vid,output:2
7.	table=0, in_port=1,dl_dst=00:00:00:00:00:01 actions=push_vlan:0x8100,set_field:4097->vlan_vid,goto_table:1
8.	table=0, in_port=2,dl_dst=00:00:00:00:00:02 actions=output:1
9.	table=1, in_port=1,dl_dst=00:00:00:00:00:01 actions=push_vlan:0x8100,set_field:4099->vlan_vid,output:2

Flow table size (*Tsize*) in penultimate hops (i.e. the Top of Rack switch) is proportional with number of groups used (*Gu*) from the total (*G*), ports per group (*Ppg*) and total number of ports of that switch (*P*):

$$Gu = \left\lceil \frac{P}{Ppg} \right\rceil \quad (2)$$

$$Tsize = 3 + (Gu + 1) * Ppg \quad (3)$$

$$Ntables = Gu + 2 \quad (4)$$

Therefore, for a 64 ports Top of rack switch, where $Ppg = 8$, $Tsize = 75$, which is an easily manageable number.

6. Conclusions

Source Routing based on VLAN tagging can easily be implemented using existing OpenFlow 1.3 functionality. This solution can be enabled by default in an entire Data Center, thus simplifying flow tables in core switches or, if packet size is an issue, only when the number of flows approaches the maximum capacity (flow resources are limited by hardware). The controller can decide when to apply

it. This may also be used to improve multipath routing, as packets can be directed on different paths from the source using fine grained distribution algorithms and switches in the core will not even be aware of it.

Also, given the fact that switching tables are static and if hardware customizations are possible, then very simple and fast hardware that only needs to support VLAN-PSSR can be built and this would provide an impressive cost reduction per switching unit.

R E F E R E N C E S

[1] *N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su.*, “Adaptive Routing Strategies for Modern High Performance Networks”, in 16th IEEE Symposium on High Performance Interconnects, 2008, pp. 165-172

[2] *F. Petrini, W.C. Feng, A. Hoisie, S. Coll and E. Frachtenberg*. “The quadrics network (qsnet): High-performance clustering technology”, in Hot Interconnects 9, 2001, pp. 125-130

[3] *M. Soliman, B. Nandy, I. Lambadaris and P. Ashwood-Smith*, "Source routed forwarding with software defined control, considerations and implications", in Proceedings of the 2012 ACM conference on CoNEXT student workshop, 2012, Nice, France

[4] *S. A. Jyothi, M. Dong and P. Godfrey*, "Towards a flexible data center fabric with source routing," in Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, Santa Clara, CA, US.

[5] *M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav and G. a. o. Varghese*, "CONGA: Distributed congestion-aware load balancing for datacenters", ACM SIGCOMM Computer Communication Review, 2014, vol. 44, no. 4, pp. 503-514

[6] *K. Agarwal, C. Dixon, E. Rozner and J. Carter*, "Shadow macs: Scalable label-switching for commodity ethernet," in Proceedings of the third workshop on Hot topics in software defined networking, 2014, Chicago, IL, US

[7] *S. Hu, K. Chen, H. Wu, W. Bai, C. Lan, H. Wang, H. Zhao and C. Guo*, "Explicit path control in commodity data centers: Design and applications," in 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), 2015, Santa Clara, CA, US.

[8] *J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah and H. Fugal*, "Fastpass: A centralized zero-queue datacenter network", ACM SIGCOMM Computer Communication Review, 2014, vol. 44, no. 4, pp. 307-318.

[9] *R. M. Ramos, M. Martinello and C. E. Rothenberg*, "SlickFlow: Resilient source routing in data center networks unlocked by OpenFlow", in Local Computer Networks (LCN), 2013 IEEE 38th Conference on, Sydney, Australia.

[10] *C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu and Y. Zhang*, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in Co-NEXT '10 Proceedings of the 6th International COnference, 2010, New York, NY, US