OPEN NETWORKING
SUMMIT 2016
MARCH 14-17, 2016 | SANTA CLARA, CA

Fix VxLAN Issue in SFC Integration by Using Eth+NSH and VxLAN-gpe+NSH Hybrid Mode

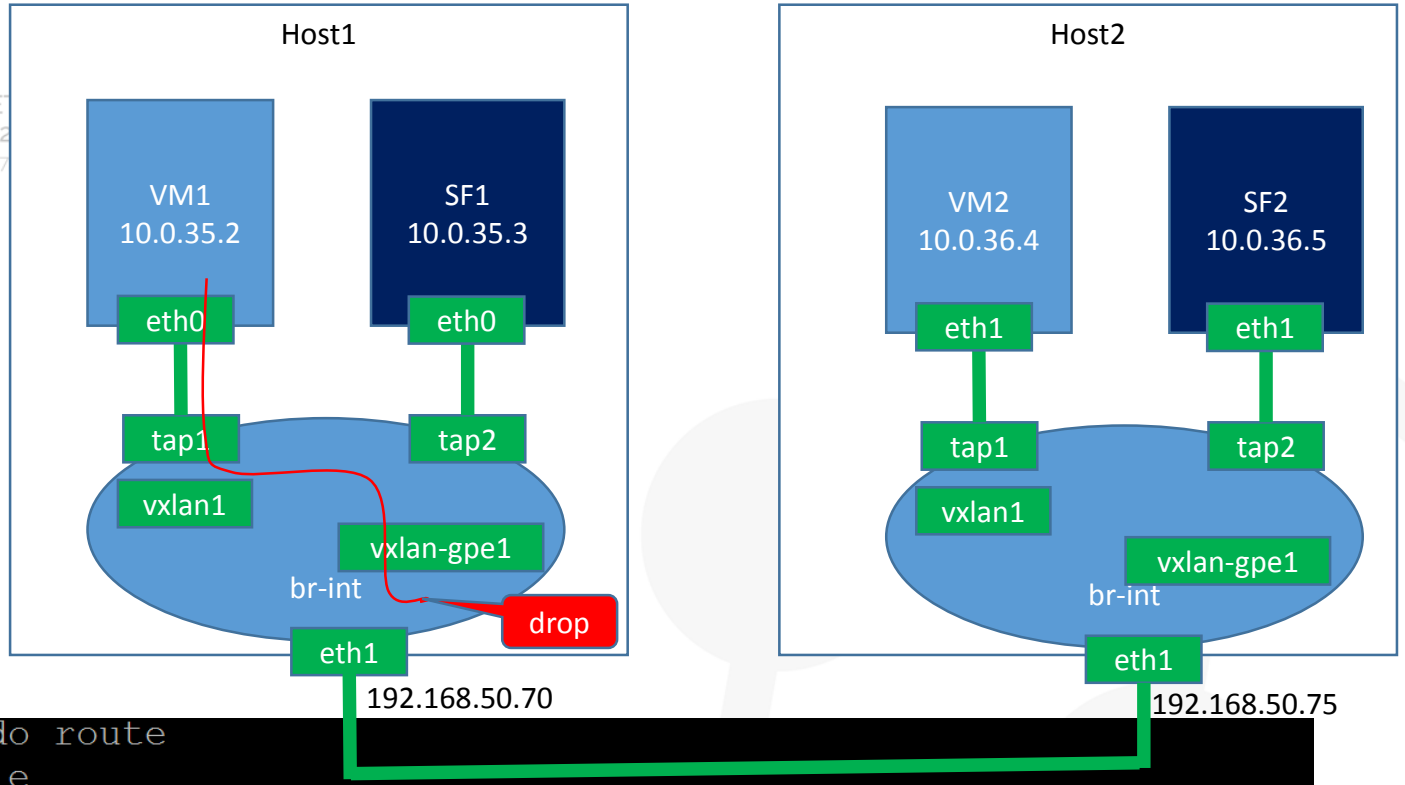
Yi Yang, Intel (yi.y.yang@intel.com)

Agenda

- VxLAN Issue in OVSDB+SFC
- How to Fix Current VxLAN issue by Eth+NSH
- Demo Introduction
- Demo
- Next Steps

VxLAN Issue in OVSDB + SFC + Openstack + Tacker

- The traffic to a SF(Service Function) has to be encapsulated by VxLAN-gpe with NSH header
- What will happen
  - Classifier sets tunnel ID (VNID), NSI (Network Service Index), NSP (Network Service Path) , tunnel destination IP
  - Output the packet to the VxLAN-gpe port
  - At this point, Linux kernel (vport_vxlan.ko) gets the control
  - vport_vxlan will query how to route this packet
  - If tunnel destination IP is set to 192.168.50.75, it will be routed to host2
  - If tunnel destination IP is set to 10.0.35.3, it will be dropped because the route table in host1 doesn't have an entry for that.
- The result: the packet with VxLAN-gpe + NSH header can't be routed to the SF




```
vagrant@gbpsfc1:~$ sudo route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         localhost       0.0.0.0         UG    0      0        0 eth0
10.0.2.0        *               255.255.255.0   U     0      0        0 eth0
172.17.0.0      *               255.255.0.0     U     0      0        0 docker0
192.168.50.0    *               255.255.255.0   U     0      0        0 eth1
vagrant@gbpsfc1:~$
```

VxLAN Issue in OVSDB+SFC

table=0,icmp,nw_dst=10.0.36.4 actions=set_nsi:255,set_nsp:0x13,set_nshc1:0xc0a83246,set_nshc2:0x9,set_nshc3:0x12345678,
set_nshc4:0x98765432,load:0xc0a8324b->NXM_NX_TUN_IPV4_DST[],load:0x9->NXM_NX_TUN_ID[0..31],output:6

192.168.50.75

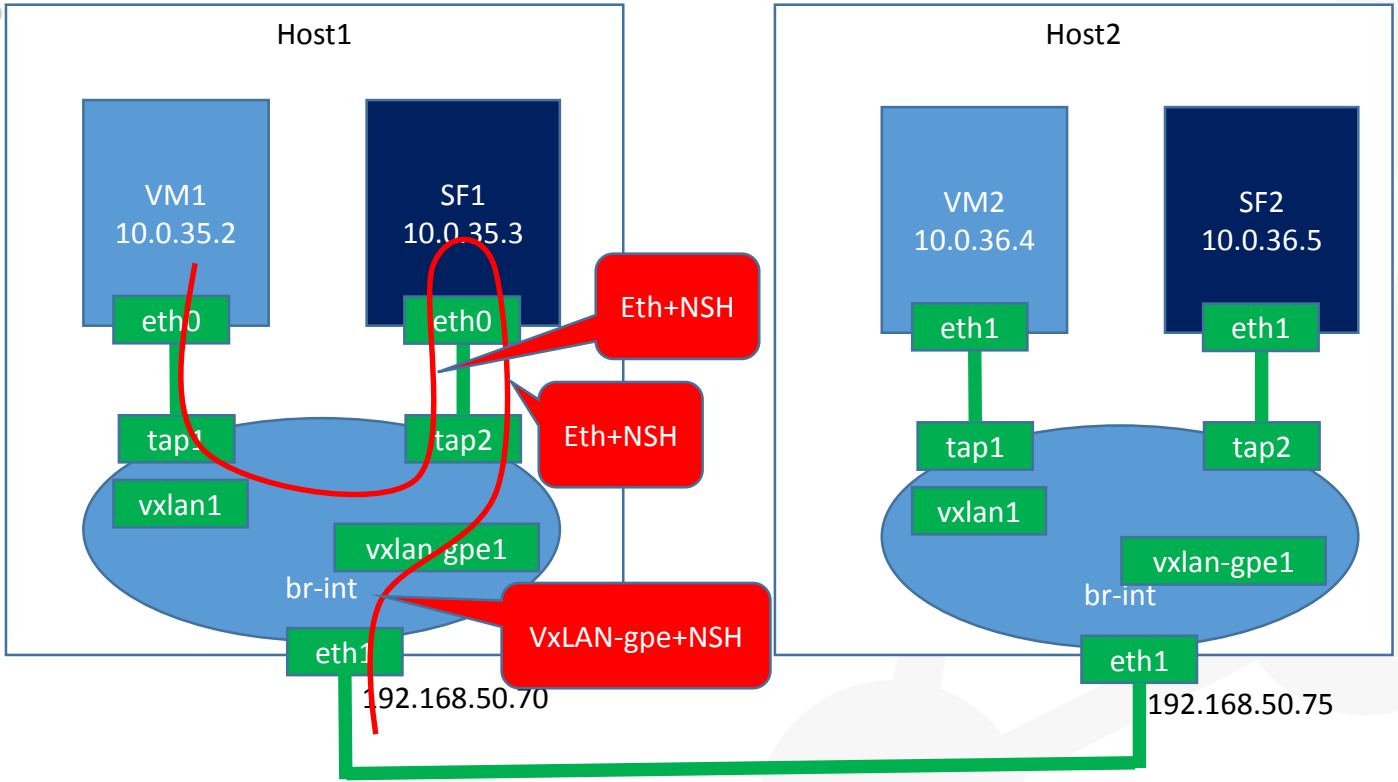
table=0,icmp,nw_dst=10.0.36.4 actions=set_nsi:255,set_nsp:0x13,set_nshc1:0xc0a83246,set_nshc2:0x9,set_nshc3:0x12345678,
set_nshc4:0x98765432,load:0x0a002303->NXM_NX_TUN_IPV4_DST[],load:0x9->NXM_NX_TUN_ID[0..31],output:6

10.0.35.3

Note: port 6 is port vxlan-gpe1

How to Fix Current VxLAN issue by Eth + NSH

- Use Eth + NSH instead of VxLAN-gpe + NSH for communication between VMs on the same compute node/ovs bridge
- Use VxLAN-gpe + NSH only for communication between VMs on different compute nodes/ovs bridges
- Need to change ovs, openflowplugin, sfc, ovsdb, gbp to support this
  - Intel owned ovs nsh support and changed ovs to support push&pop nsh&eth actions, but they are not committed into ovs git tree, Redhat and Intel are committed to submiting related changes into Linux kernel net subtree and ovs git tree
  - Openflowplugin changes: https://git.opendaylight.org/gerrit/#/c/30481/
  - Change sfcofl2 (sfc openflow renderer) and sfc classifier
  - TBD for ovsdb and gbp


Host1
VM1
10.0.35.2
SF1
10.0.35.3
eth0
eth0
Eth+NSH
Eth+NSH
tap1
tap2
vxlan1
vxlan-gpe1
br-int
VxLAN-gpe+NSH
eth1
192.168.50.70
Host2
VM2
10.0.36.4
SF2
10.0.36.5
eth1
eth1
tap1
tap2
vxlan1
vxlan-gpe1
br-int
eth1
192.168.50.75

9.2. VXLAN-gpe + NSH

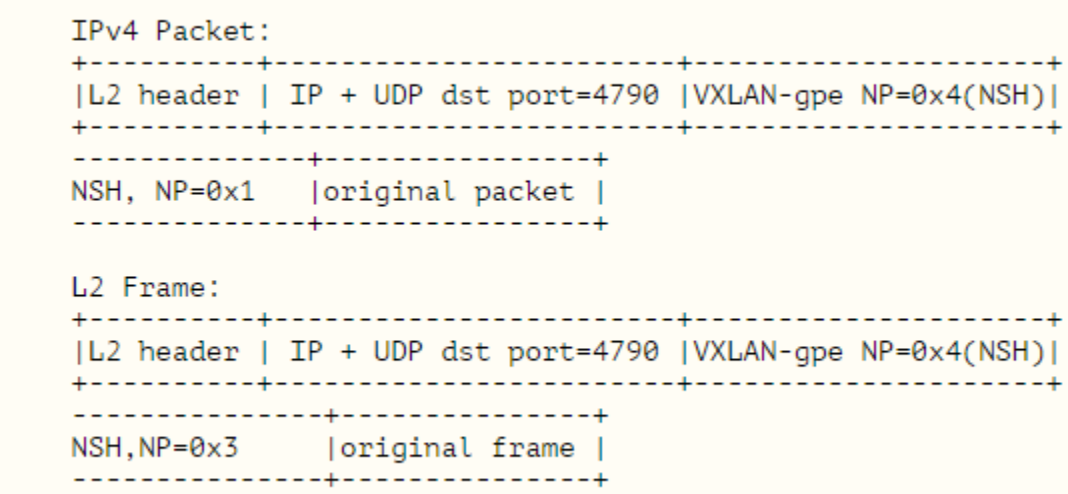


Figure 21: VXLAN-gpe + NSH

9.3. Ethernet + NSH

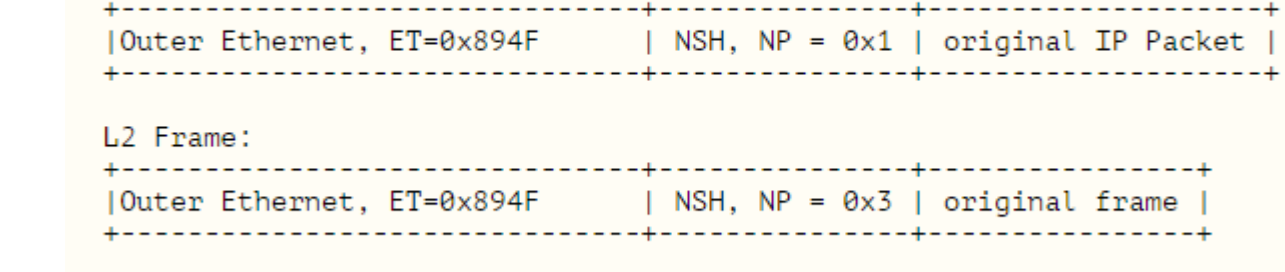


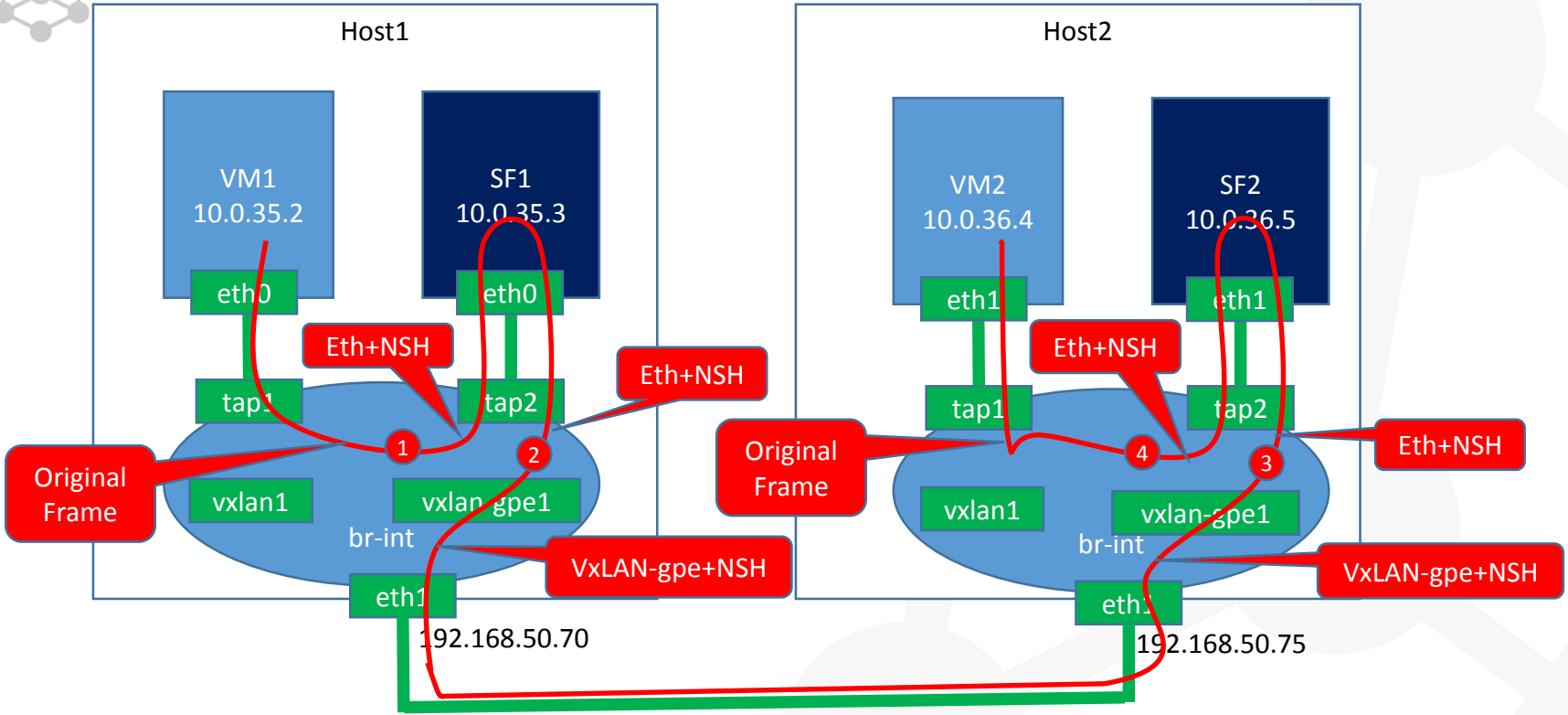
Figure 22: Ethernet + NSH

Demo Introduction

- Host1 and Host2 are vagrant boxes (VirtualBox images, Ubuntu Trusty x86_64)
- VM1, VM2, SF1 And SF2 are docker containers (Ubuntu Trusty x86_64)
- SF1 and SF2: python3 ./vxlan-tool.py --do=forward -i eth0
  - https://git.opendaylight.org/gerrit/gitweb?p=sfc.git;a=blob_plain;f=sfc-test/nsh-tools/vxlan_tool.py;h=3b05aff03aa60c556a836efc471f65d1c2595b97;hb=HEAD
- Demo two traffics
  - ICMP: ping VM2 from VM1, it will go through VM1->SF1->SF2->VM2
  - HTTP: wget http://10.0.36.4/ in VM1, it will also go through VM1->SF1->SF2->VM2

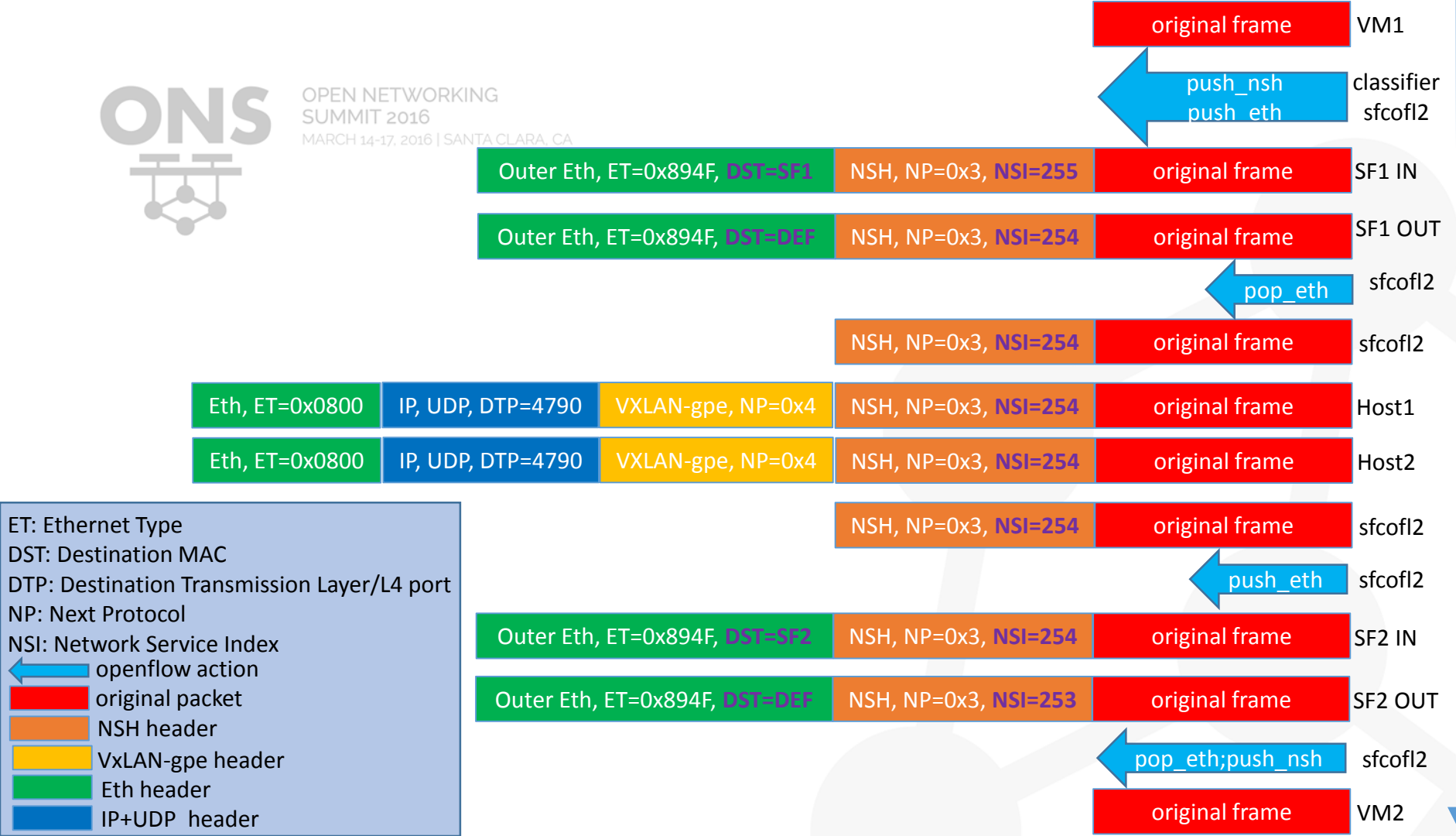
https://www.youtube.com/watch?v=3SlqKxZp9nY

1. SFC Classifier (table 0) pushes NSH header, sfcofl2 (table 4) pushes Eth header (Destination MAC is SF1's)
2. sfcofl2 (table 10) pops Eth header and sets VNID and tunnel destination IP (192.168.50.75)
3. sfcofl2 (table 4) pushes Eth header (Destination MAC is SF2's)
4. sfcofl2 (table 10) pops Eth and NSH header



Next Steps

- Commit our ovs nsh code to ovs git tree
- Push our changes in Openflowplugin (push_eth, push_nsh, pop_eth, pop_nsh) to Boron release
- Push sfc classifier and sfcofl2 changes into Boron release
- Set up the same demo in OVSDB+SFC+Openstack+Tacker integration (depends on if it is ready for such network topology)
- Push Openflowplugin Flow Programmer (A solution to openflow multi-writer issue/App coexistence issue) to Boron release
- Integrate Openflowplugin Flow Programmer to SFC, OVSDB and GBP



Host1

Classifier flow entries:

ARP response: (ping: step 1)
table=0, priority=1000,arp,arp_tpa=10.0.35.1,arp_op=1 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],
set_field:88:f0:31:b5:12:b5->eth_src,load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],
load:0x88f031b512b5->NXM_NX_ARP_SHA[],move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0xa002301->NXM_OF_ARP_SPA[],
IN_PORT

Routing: (ping: step 31, the second packet and later)
table=0, priority=1000,ip,dl_dst=88:f0:31:b5:12:b5,nw_dst=10.0.35.2 actions=set_field:88:f0:31:b5:12:b5->eth_src,
set_field:00:00:00:00:35:02->eth_dst,dec_ttl,output:2

Make sure the packets from VxLAN-gpe port and the packets with Eth+NSH header can go through sfcofl2 pipeline:
table=0, priority=1001,in_port=6 actions=load:0x9->NXM_NX_REG0[],goto_table:1 (ping: step 23)
table=0, priority=1001,encap_eth_type=20361 actions=goto_table:1 (ping: step 27)

Classifier ACL for ICMP: (ping: step 2)
table=0, priority=1000,icmp,in_port=2,nw_src=10.0.35.2,nw_dst=10.0.36.4 actions=load:0x64->NXM_NX_TUN_ID[],push_nsh,
load:0x1->NXM_NX_NSH_MDTYPE[],load:0x3->NXM_NX_NSH_NP[],load:0x4->NXM_NX_NSP[0..23],load:0xff->NXM_NX_NSI[],
load:0x1->NXM_NX_NSH_C1[],load:0x64->NXM_NX_NSH_C2[],load:0x3->NXM_NX_NSH_C3[],load:0x4->NXM_NX_NSH_C4[],
goto_table:4

Classifier ACL for TCP:
table=0, priority=1000,tcp,in_port=2,nw_src=10.0.35.2,nw_dst=10.0.36.4,tp_dst=80 actions=load:0x65->NXM_NX_TUN_ID[],
push_nsh,load:0x1->NXM_NX_NSH_MDTYPE[],load:0x3->NXM_NX_NSH_NP[],load:0x4->NXM_NX_NSP[0..23],load:0xff->NXM_NX_NSI[],
load:0x1->NXM_NX_NSH_C1[],load:0x65->NXM_NX_NSH_C2[],load:0x3->NXM_NX_NSH_C3[],load:0x4->NXM_NX_NSH_C4[],
goto_table:4

Host1

Classifier flow entries:

This flow entry is to make sure the packet returned back by VM2 can be delivered into VM1: (ping: step 31, the first packet)
table=0, priority=999,nsh_mdtype=0,encap_eth_type=0 actions=set_field:88:f0:31:b5:12:b5->eth_src,set_field:00:00:00:00:35:02->eth_dst,
output:2

Sfcofl2 flow entries:
table=0, priority=5 actions=goto_table:1

Make sure the packets from VxLAN-gpe port and the packets with Eth+NSH header can go through sfcofl2 pipeline:
table=1, priority=252,reg0=0x9 actions=goto_table:4 (ping: step 24)
table=1, priority=251,encap_eth_type=20361 actions=goto_table:4 (ping: step 28)
table=1, priority=250,ip actions=goto_table:4

table=1, priority=5 actions=drop

table=2, priority=5 actions=goto_table:3
table=3, priority=5 actions=goto_table:4

push_eth for the packets from VM1, for SF1 input: (ping: step 3)
table=4, priority=550,nsi=255,nsp=4 actions=push_eth,load:0x894f->NXM_NX_ENCAP_ETH_TYPE[],
load:0x88f031b512b5->NXM_NX_ENCAP_ETH_SRC[],load:0x3503->NXM_NX_ENCAP_ETH_DST[],goto_table:10

Set tunnel ID and tunnel destination IP for SF1 output:
table=4, priority=550,nsi=254,nsp=4 actions=move:NXM_NX_NSH_C2[]->NXM_NX_TUN_ID[0..31],load:0xc0a8324b-
>NXM_NX_TUN_IPV4_DST[],
goto_table:10

Host1

Sfcofl2 flow entries:

push_eth for the packets from SF2, for SF1 input: (ping: step 25)
table=4, priority=550,nsi=254,nsp=8388612 actions=push_eth,load:0x894f->NXM_NX_ENCAP_ETH_TYPE[],
load:0x88f031b512b5->NXM_NX_ENCAP_ETH_SRC[],load:0x3503->NXM_NX_ENCAP_ETH_DST[],goto_table:10

table=4, priority=5 actions=goto_table:10 (ping: step 29)

pop_eth and pop_nsh and resubmit to classifier for the packets from SF1 and VM2 (Reverse RSP): (ping: step 30)
table=10, priority=660,nsi=253,nsp=8388612 actions=pop_eth,pop_nsh,resubmit(,0)

Output Eth+NSH packets from VM1 to SF1/port 3: (ping: step 4)
table=10, priority=650,nsi=255,nsp=4,encap_eth_type=20361 actions=output:3

Output the Eth+NSH packets from SF2 to SF1/port 3: (ping: step 26)
table=10, priority=650,nsi=254,nsp=8388612,encap_eth_type=20361 actions=output:3

pop_eth and output the packets from SF1 to VxLAN-gpe port/port 6: (ping: step 5)
table=10, priority=650,nsi=254,nsh_mdtype=1,nsh_np=3,nsp=4,encap_eth_type=20361 actions=pop_eth,load:0x4->NXM_NX_TUN_GPE_NP[],
move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],
move:NXM_NX_NSH_C3[]->NXM_NX_NSH_C3[],move:NXM_NX_NSH_C4[]->NXM_NX_NSH_C4[],
move:NXM_NX_TUN_IPV4_DST[]->NXM_NX_TUN_IPV4_DST[],move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],output:6

table=10, priority=5 actions=drop

Host2

Classifier flow entries:

ARP response: (ping: step 15)
table=0, priority=1000,arp,arp_tpa=10.0.36.1,arp_op=1 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],
set_field:88:f0:31:b5:12:b5->eth_src,load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],
load:0x88f031b512b5->NXM_NX_ARP_SHA[],move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0xa002401->NXM_OF_ARP_SPA[],IN_PORT

Routing: (ping: step 14, the second packet and later)
table=0, priority=1000,ip,dl_dst=88:f0:31:b5:12:b5,nw_dst=10.0.36.4 actions=set_field:88:f0:31:b5:12:b5->eth_src,set_field:00:00:00:00:36:04->eth_dst,dec_ttl,output:4

Make sure the packets from VxLAN-gpe port and the packets with Eth+NSH header can go through sfcofl2 pipeline:
table=0, priority=1001,in_port=6 actions=load:0x9->NXM_NX_REG0[],goto_table:1 (ping: step 6)
table=0, priority=1001,encap_eth_type=20361 actions=goto_table:1 (ping: step 10) (ping: step 19)

Classifier ACL for ICMP: (ping: step 16)
table=0, priority=1000,icmp,in_port=4,nw_src=10.0.36.4,nw_dst=10.0.35.2 actions=load:0x66->NXM_NX_TUN_ID[],push_nsh,
load:0x1->NXM_NX_NSH_MDTYPE[],load:0x3->NXM_NX_NSH_NP[],load:0x800004->NXM_NX_NSP[0..23],load:0xff->NXM_NX_NSI[],
load:0x1->NXM_NX_NSH_C1[],load:0x66->NXM_NX_NSH_C2[],load:0x3->NXM_NX_NSH_C3[],load:0x4->NXM_NX_NSH_C4[],goto_table:4

Classifier ACL for TCP:
table=0, priority=1000,tcp,in_port=4,nw_src=10.0.36.4,nw_dst=10.0.35.2,tp_src=80 actions=load:0x67->NXM_NX_TUN_ID[],push_nsh,
load:0x1->NXM_NX_NSH_MDTYPE[],load:0x3->NXM_NX_NSH_NP[],load:0x800004->NXM_NX_NSP[0..23],load:0xff->NXM_NX_NSI[],
load:0x1->NXM_NX_NSH_C1[],load:0x67->NXM_NX_NSH_C2[],load:0x3->NXM_NX_NSH_C3[],load:0x4->NXM_NX_NSH_C4[],goto_table:4

This flow entry is to make sure the packets from VM1 can be delivered into VM2: (ping: step 14, the first packet)
table=0, priority=999,nsh_mdtype=0,encap_eth_type=0 actions=set_field:88:f0:31:b5:12:b5->eth_src,set_field:00:00:00:00:36:04->eth_dst,
output:4

Host2

Sfcofl2 flow entries:

table=0, priority=5 actions=goto_table:1

Make sure the packets from VxLAN-gpe port and the packets with Eth+NSH header can go through sfcofl2 pipeline:
table=1, priority=252,reg0=0x9 actions=goto_table:4 (ping: step 7)
table=1, priority=251,encap_eth_type=20361 actions=goto_table:4 (ping: step 11) (ping: step 20)
table=1, priority=250,ip actions=goto_table:4

sfcofl2 default flow entries:

table=1, priority=5 actions=drop
table=2, priority=5 actions=goto_table:3
table=3, priority=5 actions=goto_table:4

push_eth for the packets from SF1, for SF2 input: (ping: step 8)
table=4, priority=550,nsi=254,nsp=4 actions=push_eth,load:0x894f->NXM_NX_ENCAP_ETH_TYPE[],
load:0x88f031b512b5->NXM_NX_ENCAP_ETH_SRC[],load:0x3605->NXM_NX_ENCAP_ETH_DST[],goto_table:10

push_eth for the packets from VM2, for SF2 input:
table=4, priority=550,nsi=255,nsp=8388612 actions=push_eth,load:0x894f->NXM_NX_ENCAP_ETH_TYPE[],
load:0x88f031b512b5->NXM_NX_ENCAP_ETH_SRC[],load:0x3605->NXM_NX_ENCAP_ETH_DST[],goto_table:10 (ping: step 17)
table=4, priority=5 actions=goto_table:10 (ping: step 12)

Set tunnel ID and tunnel destination IP for SF2 output, for output to VxLAN-gpe port:
table=4, priority=550,nsi=254,nsp=8388612 actions=move:NXM_NX_NSH_C2[]->NXM_NX_TUN_ID[0..31],
load:0xc0a83246->NXM_NX_TUN_IPV4_DST[],goto_table:10 (ping: step 21)

pop_eth and pop_nsh and resubmit to classifier for the packets from SF1 and VM1 (Forward RSP):
table=10, priority=660,nsi=253,nsp=4 actions=pop_eth,pop_nsh,resubmit(,0) (ping: step 13)

Host2

Output Eth+NSH packets from SF1/VM1 to SF2/port 5: (ping: step 9)
table=10, priority=650,nsi=254,nsp=4,encap_eth_type=20361 actions=output:5

Output Eth+NSH packets from VM2 to SF2/port 5: (ping: step 18)
table=10, nsi=255,nsp=8388612,encap_eth_type=20361 actions=output:5

pop_eth and output the packets from SF2 to VxLAN-gpe port/port 6: (ping: step 22)
table=10, priority=650,nsi=254,nsh_mdtype=1,nsh_np=3,nsp=8388612,encap_eth_type=20361 actions=pop_eth,
load:0x4->NXM_NX_TUN_GPE_NP[],move:NXM_NX_NSH_C1[]->NXM_NX_NSH_C1[],
move:NXM_NX_NSH_C2[]->NXM_NX_NSH_C2[],move:NXM_NX_NSH_C3[]->NXM_NX_NSH_C3[],
move:NXM_NX_NSH_C4[]->NXM_NX_NSH_C4[],move:NXM_NX_TUN_IPV4_DST[]->NXM_NX_TUN_IPV4_DST[],
move:NXM_NX_TUN_ID[0..31]->NXM_NX_TUN_ID[0..31],output:6

table=10, priority=5 actions=drop