Open Hardware and Software for networking

SONiC NOS evaluation

Łukasz Makowski
makowski@uva.nl

Paola Grosso
pgrosso@uva.nl

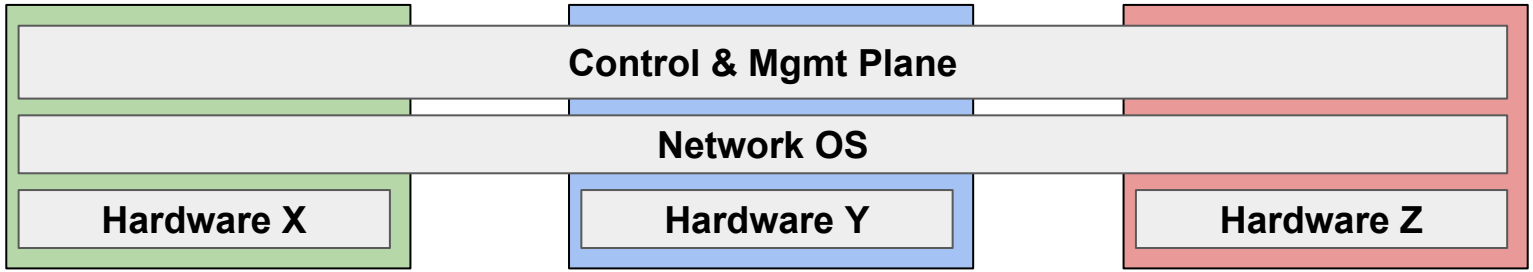
UNIVERSITEIT VAN AMSTERDAM

System and Network Engineering

SURF NET

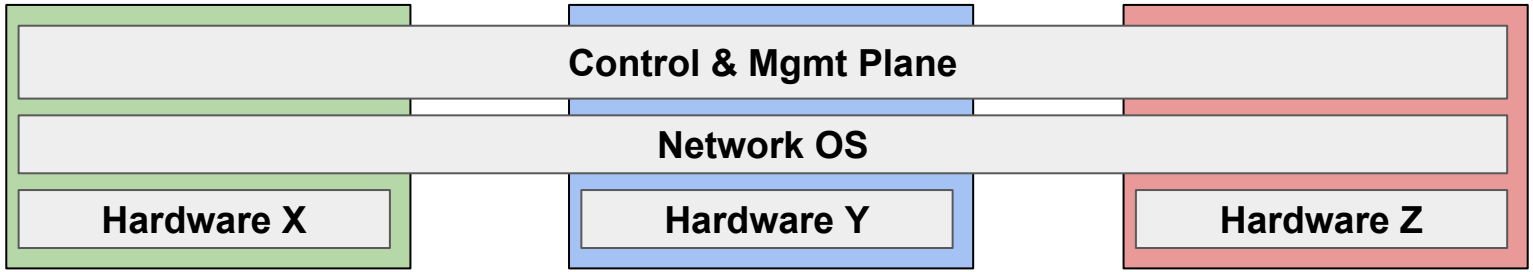
Open networking

- Cost reduction
- No vendor lock-in
- Common NOS → unified management



Open networking: it’s a fact

- LINX: https://www.edge-core.com/news-inquiry.php?id=302
- NTT: https://cumulusnetworks.com/customers/ntt/
- Cisco working on getting IOS XR running with OCP hardware



Network Operating Systems (NOS)

- There is a variety of vendors offering white-box network hardware with the combination with commercial NOS, such as:
  - Cumulus Linux
  - IPinfusion OcNOS
  - PicaOS
  - …
- These OSes are composed of closed source components limiting the user's freedom and extensibility
  - It is beneficial to have an open-source alternative

- Are there any community supported open-source NOS?

Open-source NOS

- Open Network Linux (ONL)
  - Extensive platform support but not a complete NOS
- Azure SONiC
  - Extensive platform support
  - Focused on cloud environments (limited feature-set)
- Open Switch (OPX)
  - Almost only Dell networking hardware
  - Feature-set focused on general use cases

Research questions

1. How feasible is to use SONiC outside of cloud environment?
    a. Can one add new features/protocols
    b. Is there a “grey” area? i.e. features not officially supported but working without a great effort
2. How seamless/transparent is the multi-platform support?

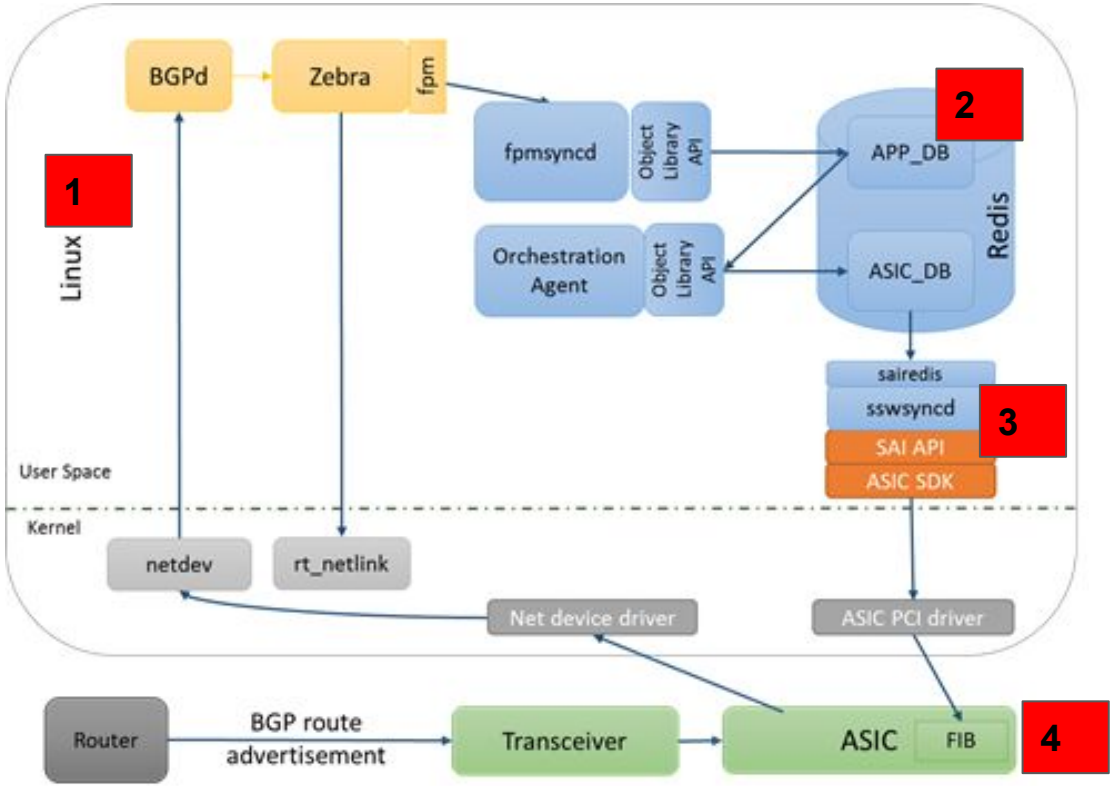
Switch Abstraction Interface

- One API to rule them all (ASICs)

  https://github.com/opencomputeproject/SAI

  SAI is the initiative to create a vendor independent ASIC programming API

SONiC architecture: life of routing entry



https://github.com/Azure/SONiC/wiki/Architecture

```
root@sonic:~# ip addr add db8:dead:beef:0001::1/64 dev Ethernet20
root@sonic:~# ip -6 route show
db8::/64 dev Ethernet20  proto kernel  metric 256
db8:dead:beef::/64 dev Ethernet20  proto kernel  metric 256
db8:dead:beef:1::/64 dev Ethernet20  proto kernel  metric 256
fe80::/64 dev Bridge  proto kernel  metric 256
fe80::/64 dev Ethernet20  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
root@sonic:~#
```

1

```
root@sonic:/var/log/swss# redis-cli -n 1 monitor | grep 'ROUTE_TABLE:db8:dead:beef'
1543766033.284003 [0 unix:/var/run/redis/redis.sock] "EVALSHA" "213b55cf0c203f65f4fc5ced481cb229dfbdc7ba" "4" "ROUTE_TABLE_CHANNEL" "ROUTE_TABLE_KEY_SET" "ROUTE_TABLE:db8:dead:beef:1::/64" "ROUTE_TABLE:db8:dead:
beef:1::/64" "G" "db8:dead:beef:1::/64" "nexthop" "" "ifname" "Ethernet20"
1543766033.284441 [0 lua] "HSET" "ROUTE_TABLE:db8:dead:beef:1::/64" "nexthop" ""
1543766033.284536 [0 lua] "HSET" "ROUTE_TABLE:db8:dead:beef:1::/64" "ifname" "Ethernet20"
1543766033.286328 [0 lua] "HGETALL" "ROUTE_TABLE:db8:dead:beef:1::/64"
```

2

```
root@sonic:/var/log/swss# tail -f sairedis.rec | grep SAI_OBJECT_TYPE_ROUTE_ENTRY
2018-12-02.15:53:53.276354|c|SAI_OBJECT_TYPE_ROUTE_ENTRY:{"dest":"db8:dead:beef:1::/64","switch_id":"oid:0x21000000000000","vr":"oid:0x3000000000023"}|SAI_ROUTE_ENTRY_ATTR_PACKET_ACTION=SAI_PACKET_ACTION_FORWARD
|SAI_ROUTE_ENTRY_ATTR_NEXT_HOP_ID=oid:0x6000000000608
2018-12-02.15:53:53.279823|c|SAI_OBJECT_TYPE_ROUTE_ENTRY:{"dest":"db8:dead:beef:1::1/128","switch_id":"oid:0x21000000000000","vr":"oid:0x3000000000023"}|SAI_ROUTE_ENTRY_ATTR_PACKET_ACTION=SAI_PACKET_ACTION_FORWA
RD|SAI_ROUTE_ENTRY_ATTR_NEXT_HOP_ID=oid:0x1000000000001
```

3

```
Unit 0, Total Number of IPv6 entries: 196608
Max number of ECMP paths 64
Free IPv6 entries available: 196596
#     VRF     Net addr                                  Next Hop Mac        INTF MODID PORT PRIO CLASS HIT VLAN
65552 0        0db8:0000:0000:0000:0000:0000:0000:0000/64  00:00:00:00:00:00 100003    0     0     0    1 n
17    0        0db8:0000:0000:0000:0000:0000:0000:0002/128  00:00:00:00:00:00 100003    0     0     0    1 n
65553 0        0db8:dead:beef:0000:0000:0000:0000:0000/64  00:00:00:00:00:00 100003    0     0     0    1 n
18    0        0db8:dead:beef:0000:0000:0000:0000:0001/128  00:00:00:00:00:00 100003    0     0     0    1 n
65554 0        0db8:dead:beef:0001:0000:0000:0000:0000/64  00:00:00:00:00:00 100003    0     0     0    1 n
19    0        0db8:dead:beef:0001:0000:0000:0000:0001/128  00:00:00:00:00:00 100003    0     0     0    1 n
16    0        0000:0000:0000:0000:0000:0000:0000:0000/0  00:00:00:00:00:00 100002    0     0     0    0 n

drivshell>
root@sonic:~#
```

4

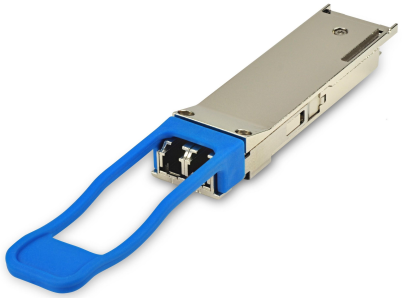
Experiments

1. Pluggables support
2. ACL support
3. Including extra features
    a. OSPF (Open Shortest Path First) with FRR (Free Range Routing)
    b. STP (Spanning Tree Support)

Mellanox SN2100 (Mellanox Spectrum)

Arista 7050QX-32S (Broadcom Trident2)

Pluggables support



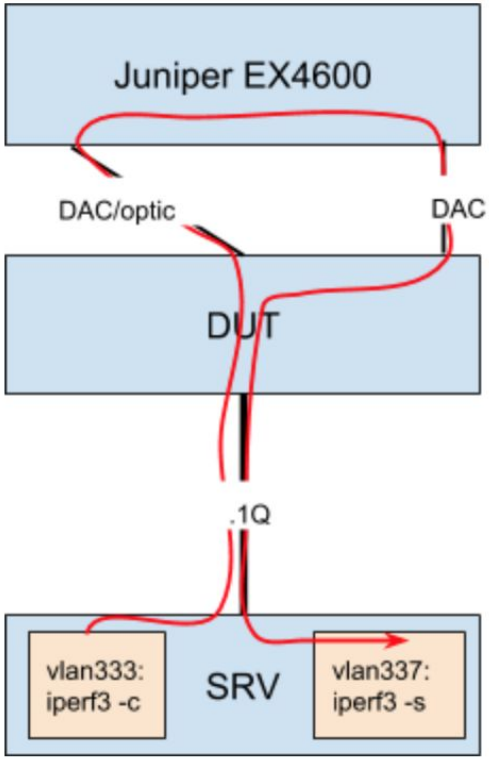
QSFP+ optics

DAC breakout cable

DAC

QSA

Pluggables support: Method & Results



Type	Arista	Mellanox
40G DAC	OK	OK
100G DAC	N/A	OK
40G LR4 optic	OK	OK
100G LR4 optic	N/A	OK
4x10G breakout	CAPABLE	OK
10G LR4 optic+ QSA	CAPABLE	OK

Pluggables support: breakout and QSA adapters

- From the configuration point of view breakout-cable and QSA adapter does not differ too much. Both are about splitting/filtering-out the lanes of QSFP port.
- Mellanox: hence the configuration of breakout was supported, we managed to use QSA in the similar way
- Arista/Broadcom: neither is supported, however it must be possible (as commercial NOSes support it). The work around is to bypass SAI and try adjusting the ASIC directly.

Pluggables support: breakout and QSA adapters (cont.)

```
root@sonic:/etc/bcm# head -20 /etc/bcm/td2-a7050-qx32s-32x40G.config.bcm | grep .
#/*****************************************************************************
# *
# *      File:   config.bcm.clearlake (7050-QX32S)
# *      Name:
# *
# *      Description: This file contains SDK properties for an Arista
# *      Clearlake platform.
# *
# *----------------------------------------------------------------------------
# *****************************************************************************/
#####################################################################
# BCM Config file for Clearlake platform
#  - 32x40g Portmode
# ALPM enable
l3_alpm_enable=2
ipv6_lpm_128b_enable=1
l2_mem_entries=32768
l3_mem_entries=16384
```

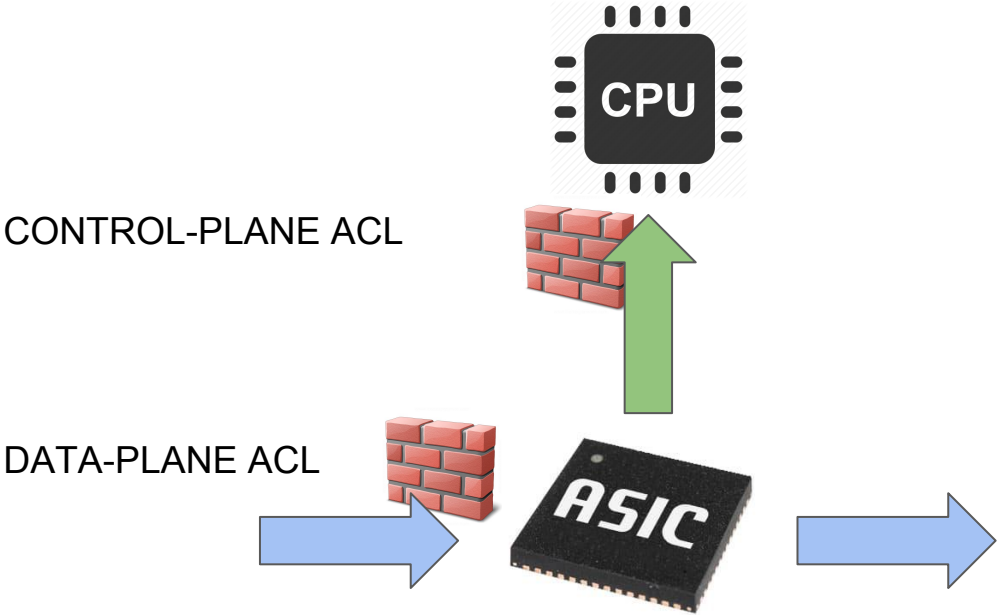
Pluggables support: breakout and QSA adapters (cont.II)

```
root@sonic:/etc/bcm# grep  '_5\.' /etc/bcm/td2-a7050-qx32s-32x40G.config.bcm --color
phy_an_c37_5.0=3
phy_an_c73_5.0=0
phy_xaui_rx_polarity_flip_5.0=0x0
phy_xaui_tx_polarity_flip_5.0=0x0
port_init_autoneg_5.0=0
port_phy_addr_5.0=0x7f
portmap_5.0=13:40
serdes_firmware_mode_5.0=2
xgxs_rx_lane_map_5.0=0x0321
xgxs_tx_lane_map_5.0=0x0321
serdes_pre_driver_current_lane0_5.0=0x7
serdes_driver_current_lane0_5.0=0x7
serdes_preemphasis_lane0_5.0=0xc2f0
serdes_pre_driver_current_lane1_5.0=0x7
serdes_driver_current_lane1_5.0=0x7
serdes_preemphasis_lane1_5.0=0xc2f0
serdes_pre_driver_current_lane2_5.0=0x7
serdes_driver_current_lane2_5.0=0x7
serdes_preemphasis_lane2_5.0=0xc2f0
serdes_pre_driver_current_lane3_5.0=0x7
serdes_driver_current_lane3_5.0=0x7
serdes_preemphasis_lane3_5.0=0xc2f0
```

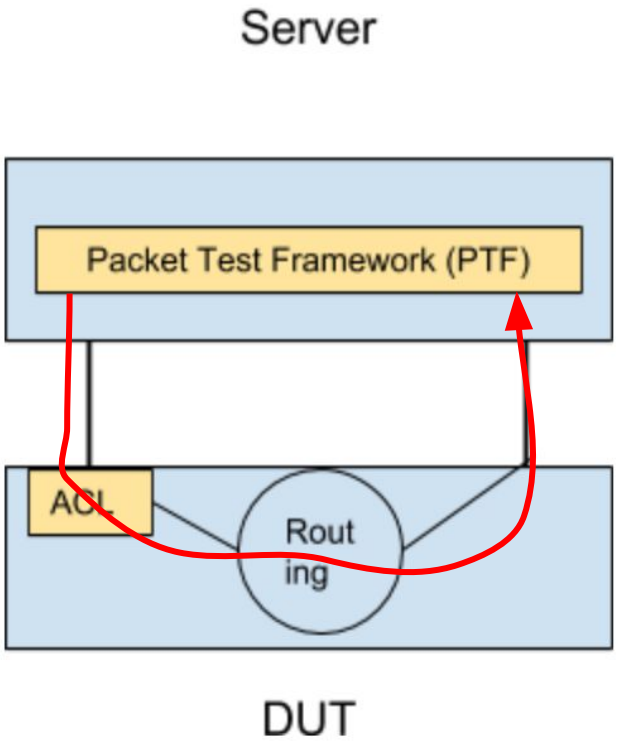
Parameters manual:

https://broadcom-switch.github.io/OpenNSL/doc/html/OPENNSL_CUSTOMIZING_OPENNSL.html

ACL support: Types of ACLs



Data-plane ACL support: Method



- Adapted SONIC testing framework available in the sonic-mgmt repository (https://github.com/Azure/sonic-mgmt)

1. ACL config is applied on the DUT
2. PTF generates a packet which is destined to the second interface of a server
3. Depending on the scenario, it is expected the packet will or will not arrive

Data-plane ACL support (cont.)

TEST TYPE	ARISTA	MELLANOX
Verify source IP match	IPv4 + IPv6	IPv4 only
Verify destination IP match		
Verify L4 source port match		
Verify L4 destination port match		
Verify ip protocol match		
Verify TCP flags match		
Verify source port range match		
Verify destination port range match		
Verify rules priority		
Verify IP protocol & source IP match		
Verify source IP match - UDP packet and UDP protocol		

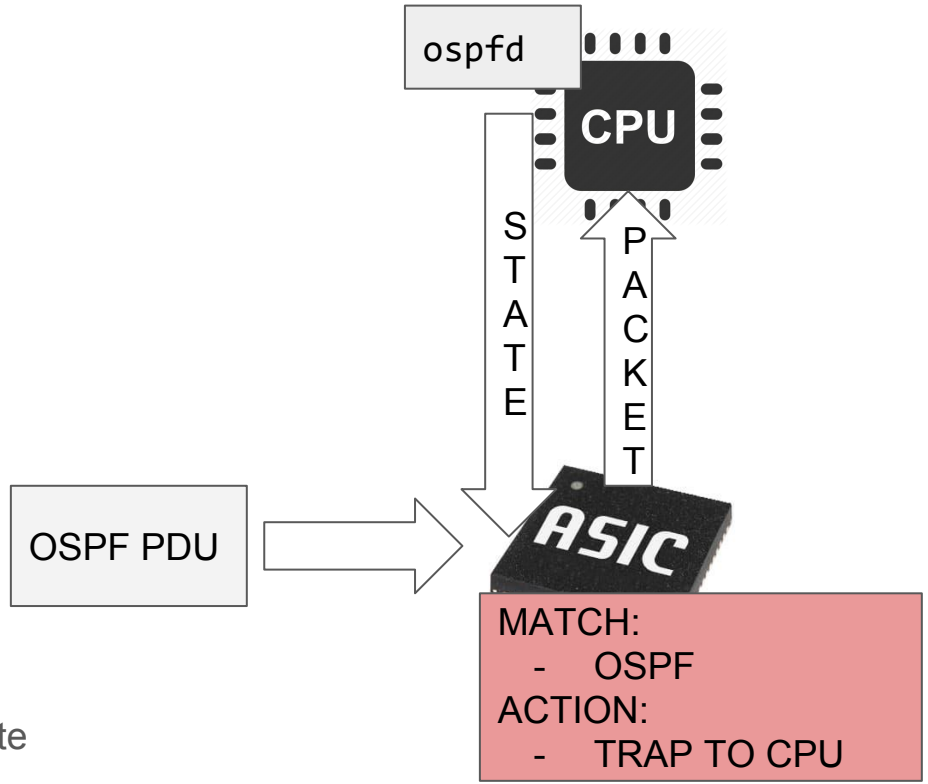
Control-plane ACL

Type	Arista	Mellanox
SSH	IPv4+IPv6	IPv4+IPv6
SNMP	IPv4+IPv6	IPv4+IPv6
NTP	n/a	IPv4+IPv6

- Arista ACL were based on sshd and snmpd daemons configuration features
- Mellanox ACL are converted to iptables rules
- Both platforms supported filtering based on the source IPv4 and IPv6 addresses

Extensibility

- Open-source NOS should allow adding new features required in the specific use-case
- For SONiC, in case of adding a new protocol we foresee the following steps:
  a. PDU CPU trap. Protocol-specific packets/frames trapping towards CPU.
  b. Control-plane application. A control-plane application implementing given protocol.
  c. State syncer. Component syncing the state a control-plane application calculated against the ASIC-attached ports.



Extensibility: OSPF with FRR

A. Configuring CPU TRAP

```
{
     "COPP_TABLE:trap.group.ospf": {
         "trap_ids": "ospf",
         "trap_action":"trap",
         "trap_priority":"10",
         "queue": "4"        },
         "OP": "SET"
 }
```

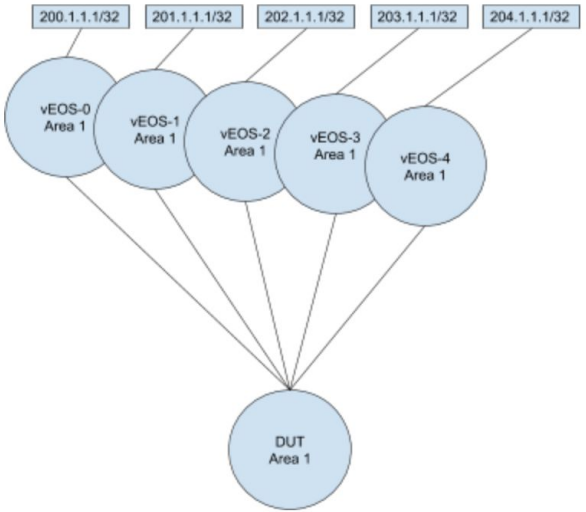
B. Deploying FRR + enabling OSPF

- FRR has already been an optional component of SONiC. We compiled a new OS image including it.
- Enabling ospfd processes is trivial.

Extensibility: OSPF with FRR (cont.)

C. Syncer

- The routes computed by ospfd need to be programmed in the ASIC
- Fortunately, SONiC already implements such for the purpose of bgpd




```
admin@sonic:~$ vtysh -c 'sh ip rou'
#output omitted
O>* 200.1.1.1/32 [110/20] via 10.33.0.2, Vlan333, 2d02h53m
O>* 201.1.1.1/32 [110/20] via 10.34.0.2, Vlan334, 2d02h54m
O>* 202.1.1.1/32 [110/20] via 10.35.0.2, Vlan335, 2d02h53m
O>* 203.1.1.1/32 [110/20] via 10.36.0.2, Vlan336, 2d02h53m
O>* 204.1.1.1/32 [110/20] via 10.37.0.2, Vlan337, 2d02h53m
```

Extensibility: STP

A. Configuring CPU TRAP

```
{
  "COPP_TABLE:trap.group.stp": {
      "trap_ids": "stp",
      "trap_action":"trap",
      "trap_priority":"4",
      "queue": "4"
  },
  "OP": "SET"
}
```

- Mellanox: no problem, can observe the STP packets with tcpdump :-)
- Arista/Broadcom: :-(

```
syncd_main: Runtime error: :- processEvent: failed to execute api: create, key:
SAI_OBJECT_TYPE_HOSTIF_TRAP:oid:0x220000000005e5, status: SAI_STATUS_NOT_IMPLEMENTED
```

Extensibility: STP (cont.)

B. STP application: linux-bridge, mstpd (user-space STP daemon)

C. Syncing STP calculated state back to the ASIC

- No piggybacking anymore, would need to develop the syncer
- Verify if SAI adapters from Broadcom and Mellanox support SAI STP API:
  - Mellanox :-)
  - Broadcom :-(

Conclusions

- Despite its focus on cloud environments, we believe that it has the potential to be used in regular data-center networks. Nevertheless, SONIC is definitely not a feature-rich NOS comparable with major vendor software.

- SAI provides a significant degree of flexibility while implementing the ASIC control-code working over multiple vendors.
  - Commonly used features can be expected to be well tested and work without issues.
  - The differences between the SAI specification and what is actually implemented by vendors exist e.g. lack of port breakout support or STP API on Arista platform.

Future work

- Cooperation with Astron on using SAI/P4 programmable devices in SKA/LOFAR system
- Analysis of new products:
  - New open NOS projects announced
    - Stratum by Google
    - DANOS by AT&T
  - New ASIC vendors
    - Marvell
    - Centec
    - Nephos