CERN

$P4_{flow}$: A software-defined networking approach with programmable switches for accounting and forwarding IPv6 packets with user-defined flow label tags

CERN
IT Department CS Group

26th International Conference on Computing in High Energy & Nuclear Physics (CHEP23)

Carmen Misa Moreira
Edoardo Martelli

Outline

Background

- Programming protocol-independent packet processors: P4 language
- EdgeCore Wedge100BF-32QS
- Network Operating System
- Packet and flow marking specification

Accounting and forwarding

- First approach: layer 3
- Second approach: layer 2
- Demo SC22

Routing

- MultiONE proposal
- GÉANT P4Lab
- MultiONE testbed in GP4Lab

Conclusions and future work

Background

Programming protocol-independent packet processors: P4 language

Language for programming the data plane of network devices

- Define how packets are processed
- P4 program structure: header types, parser/deparser, match-action tables, user-defined metadata and intrinsic metadata

Domain-specific language designed to be implementable on a large variety of targets

- Programmable network interface cards, FPGAs, software switches and hardware ASICs.

EdgeCore Wedge100BF-32QS

- 100GbE Data Center Switch
  - Bare-Metal Hardware
  - L2/L3 Switching
  - 32xQSFP28 Ports
- Data-Plane Programmability
  - Intel Tofino Switch Silicon
  - Barefoot Networks
- Quad-Pipe Programmable Packet Processing Pipeline
  - 6.4 Tbps Total Bandwidth
- CPU: Intelx86 Xeon 2.0GHz
  - 8-core/48GB/2TB SSD

Intel Tofino P4-programmable Ethernet Switch ASIC

EdgeCore Wedge100BF-32QS

Network Operating System

RARE/FreeRtr

- Controls the data plane by managing entries in routing tables
- Free and open source router operating system
- Export forwarding tables to DPDK or hardware switches
  - via OpenFlow or P4lang
- No global routing table
  - Every routed interface must be in a virtual routing table

Packet and flow marking specification

Flow label field of IPv6 header: 20 bits

- 5 entropy bits to match RFC 6436
- 9 bits to define the science domain
- 6 bits to define the application/type of traffic

Scitags

- Scientific network tags initiative [1]

Bits 12 - 13 Entropy	Bits 14 - 22 Science Domain	Bit 23 Entropy	Bits 24 - 29 Application	Bits 30 - 31 Entropy

Astro/HEP Science Domains:

```
Reserved - 0
Default  - 65536
ATLAS    - 32768
CMS      - 98304
LHCb     - 16384
ALICE    - 81920
Bellell  - 49152
SKA      - 114688
LSST     - 73728
DUNE     - 8192
```

Application:

```
Reserved            - 0
Default             - 1
perfSONAR           - 2
Cache               - 3
DataChallege        - 4
AnalysisDownload    - 9
DataAccess          - 10
CLIDownload         - 13
ProductionDownload  - 19
```

Shawn McKee (University of Michigan)

CHEP23 Talk: 8th May 2023, 15:00

Identifying and Understanding Scientific Network Flows

Accounting and forwarding

First approach: layer 3

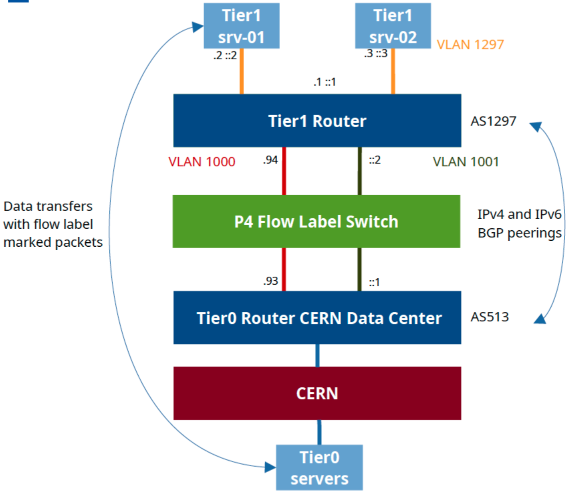
Network configuration:

- Virtual Routing Forwarding
- Policy-based routing based on flow label field value
  - Flow label 10 → VLAN 40
  - Flow label 20 → VLAN 41
- SRV-01 managed by Cisco TRex Realistic Traffic Generator
  - Python script Scapy library: generate IPv6 packets flow label tagged
  - Cisco TRex Client: Python script → Scapy library
  - Cisco TRex Server: get statistic of the traffic in real-time
- SRV-02 managed by DPDK FreeRtr

Second approach: layer 2

Network configuration:

- Emulates a Tier 1/0 link
- Tier1/0 routers
  - IPv4/IPv6 BGP peerings
- Tier0 router
  - LHCOPN production border router
- Pure layer 2 bridges
  - VLAN 1000: IPv4 traffic
  - VLAN 1001: IPv6 traffic
- Tier0 servers
  - OpenStack production servers



Second approach: layer 2

P4 switch network configuration: pure layer 2 bridges and access-list

```
access-list acl_all_ipv6_flowlabels
    # Match <Experiment> and <DataAccess Application>
    sequence 10 permit all any all any all flow 131076 & 163880
    sequence 11 permit all any all any all flow 65540  & 163880
    sequence 12 permit all any all any all flow 49152  & 163880
    sequence 13 permit all any all any all flow 114688 & 163880
    # Match <Experiment> and <perfSONAR Application>
    sequence 20 permit all any all any all flow 131072 & 261632
    sequence 21 permit all any all any all flow 65536  & 261632
    sequence 22 permit all any all any all flow 49152  & 261632
    sequence 23 permit all any all any all flow 114688 & 261632
    # Permit the rest of the traffic
    sequence 30 permit all any all any all
    exit
```

```
ATLAS   <DataAccess>
CMS     <DataAccess>
BelleII <DataAccess>
SKA     <DataAccess>
```

```
ATLAS   <perfSONAR>
CMS     <perfSONAR>
BelleII <perfSONAR>
SKA     <perfSONAR>
```

```
interface sdn1.1000
    description [VLAN ID=1000]
    bridge-group 1
    no shutdown
    no log-link-change
    exit
```

VLAN 1000 belongs to bridge 1

```
interface sdn1.1001
    description [VLAN ID=1001]
    bridge-group 2
    bridge-filter ipv6in acl_all_ipv6_flowlabels
    no shutdown
    no log-link-change
    exit
```

VLAN 1001 belongs to bridge 2
Filter IPv6 traffic at the input based on the access-list sentences

Second approach: layer 2

IPv6 packets flow label tagged were generated by using:

- iperf3
- ipv6_flow_label library developed by Marian Babik (CERN)
- eBPF_flow_label library developed by Tristan Sullivan (University of Victoria)

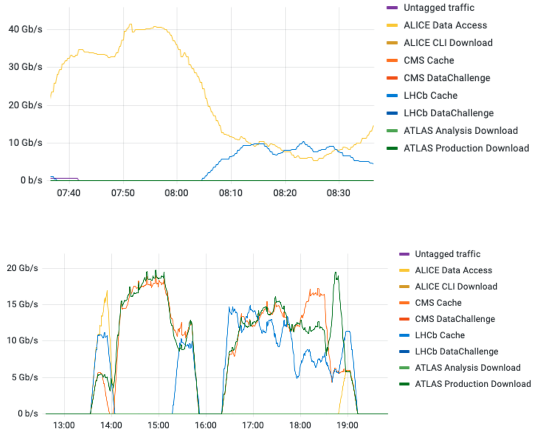
```
E513-E-YECWH-1#show access-list acl_all_ipv6_flowlabels
seq  txb  txp  rxb            rxp         last      timout    cfg
10   0+0  0+0  0+12374638771  0+8743031   00:03:02  00:00:00  permit all any all any all flow 131076&163880   ATLAS   <DataAccess>
11   0+0  0+0  0+37019728635  0+24984028  00:02:30  00:00:00  permit all any all any all flow 65540&163880    CMS     <DataAccess>
12   0+0  0+0  0+23940164205  0+15797973  00:02:00  00:00:00  permit all any all any all flow 49152&163880    BelleII <DataAccess>
13   0+0  0+0  0+18150017192  0+12017039  00:02:00  00:00:00  permit all any all any all flow 114688&163880   SKA     <DataAccess>

20   0+0  0+0  0+30346726207  0+20005622  00:01:29  00:00:00  permit all any all any all flow 131072&261632   ATLAS   <perfSONAR>
21   0+0  0+0  0+25281078379  0+16663278  00:01:29  00:00:00  permit all any all any all flow 65536&261632    CMS     <perfSONAR>
22   0+0  0+0  0+28556351375  0+19008806  00:00:58  00:00:00  permit all any all any all flow 49152&261632    BelleII <perfSONAR>
23   0+0  0+0  0+37078713993  0+25770785  00:00:26  00:00:00  permit all any all any all flow 114688&261632   SKA     <perfSONAR>
30   0+0  0+0  0+2715536713   0+1802921   00:00:26  00:00:00  permit all any all any all
```

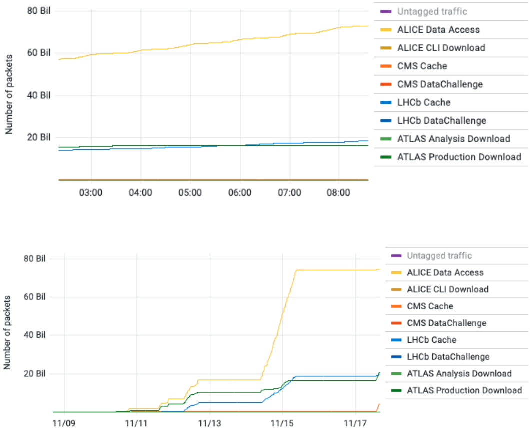
Counters of the access-list on the P4 switch

Demo SC22

- We demonstrated the accounting of tagged packets is feasible.



Counters of an access-list in bits/s



Counters of an access-list in number of packets

Routing

MultiONE proposal

Separate the traffic into different VPNs based on the IPv6 flow label value.

- MultiONE network: 3 VPNs (blue, green, red)

  + a default VPN for IPv4 and untagged traffic.

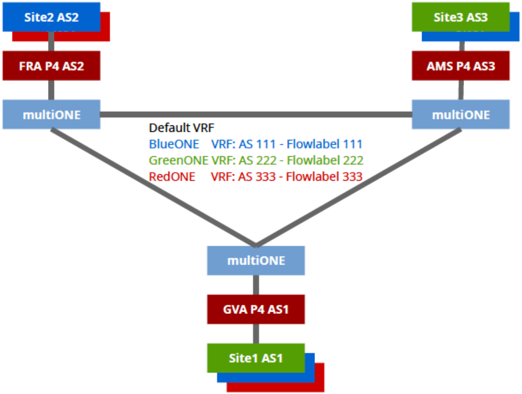
  - COTS routers with BGP and IPv6.
  - Peering with the site routers and redistribute the received prefixes.

- P4 site routers: to access the proper multiONE VPN based on the routes received from BGP and

  flow label tag of the packets.

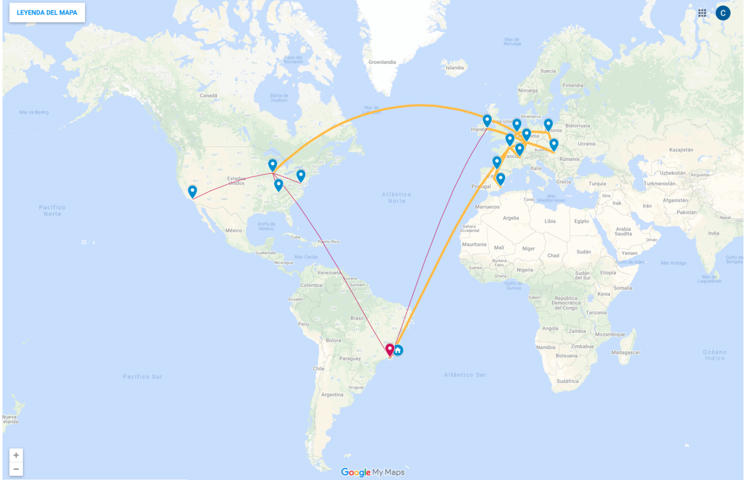
  - P4 programmable switches [P4Lab].
  - Announce the IPv6 prefixes of the local servers to the connected VRFs via BGP.

- Site servers: generate and receive tagged traffic.

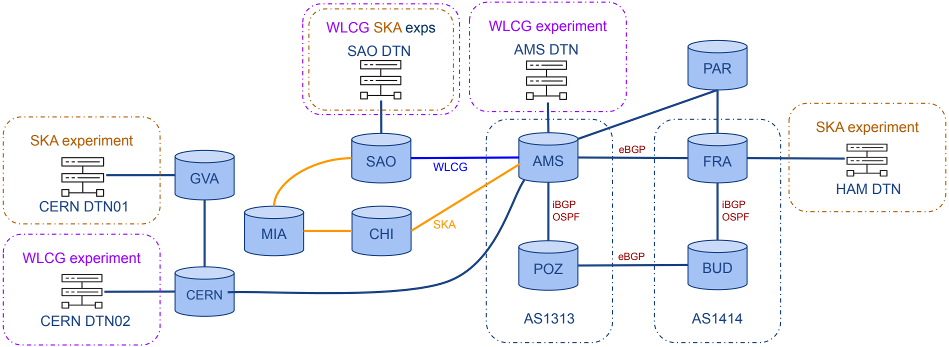


MultiONE testbed.

GÉANT P4Lab

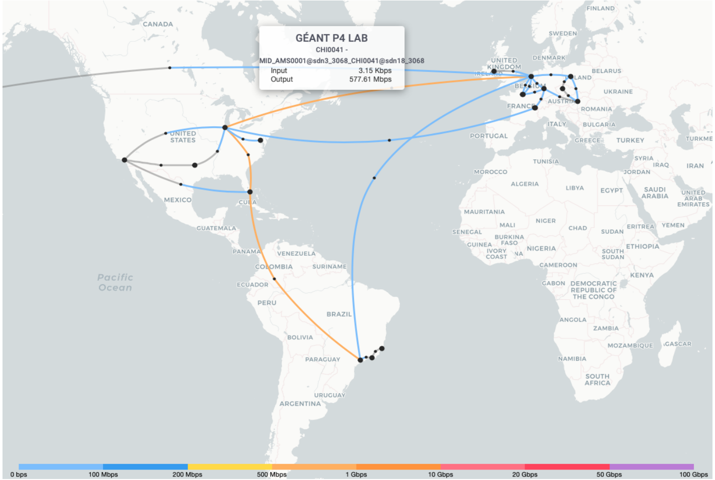


MultiONE testbed in GP4Lab



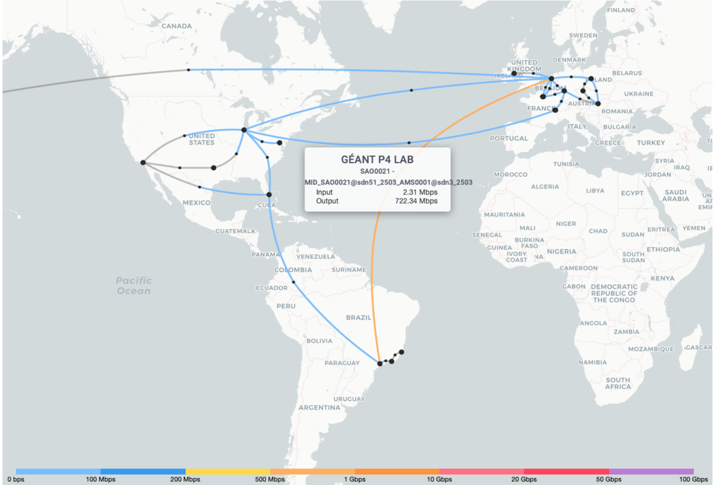
- SAO P4 switch routes the traffic with PBR rules based on an access-list.
  - WLCG traffic routing: SAO DTN → SAO → AMS → AMS DTN
  - SKA traffic routing: SAO DTN → SAO → MIA → CHI → AMS → FRA → FRA DTN
- CERN DTNs generates tagged traffic to AMS DTN and HAM DTN.
  - The traffic is routed in the squared topology to WLCG or SKA VPN so that LHCONE sites can only access other sites belonging to the same experiment and organization.

MultiONE testbed in GP4Lab



SKA traffic routing from São Paulo to Amsterdam via Chicago and Miami.

MultiONE testbed in GP4Lab



WLCG traffic routing from São Paulo directly to Amsterdam.

Conclusions and future work

- The IPv6 flow label accounting and forwarding can be implemented at layer 3 and layer 2. It was demonstrated at SC22.
- By using the GP4Lab we demonstrated that MultiONE can be implemented by using PBR rules based on an access-list with the flow label definitions on the clients to control the access to each VPN.

Thanks for your attention!

CERN