Research Networking Technical WG Update

Shawn McKee / University of Michigan and Marian Babik / CERN

#49 LHCONE/LHCOPN Meeting

(https://indico.cern.ch/event/1146558/)

Oct 24-25, 2022

History

- HEPiX Network Functions Virtualisation Working Group
  - Working Group Report was published at the end of 2019 with three chapters
    - Cloud Native DC Networking
    - Programmable Wide Area Networks
    - Proposed Areas of Future Work
- LHCOPN/LHCONE workshop (spring 2020)
  - Requirements on networks from the WLCG experiments
- Research Networking Technical Working Group
  - Formed after the workshop in response to the requirements discussion
  - 98 members from ~ 50 organisations have joined
  - Three main areas of work:
    - **Network Visibility: Packet and Flow Marking** - viewed as the appropriate first step; regular meetings every ~2 months since summer 2020
      - Packet Marking Document
        - Outlines available technologies, standards and stakeholders perspectives
        - This has led to Scientific Network Tags (scitags) initiative, which is presented today
    - **Traffic Shaping -** Using techniques like packet pacing to achieve consistent throughput.
    - **Network Orchestration** - followed up by GNA-G, SENSE and FABRIC

Network Visibility Motivation

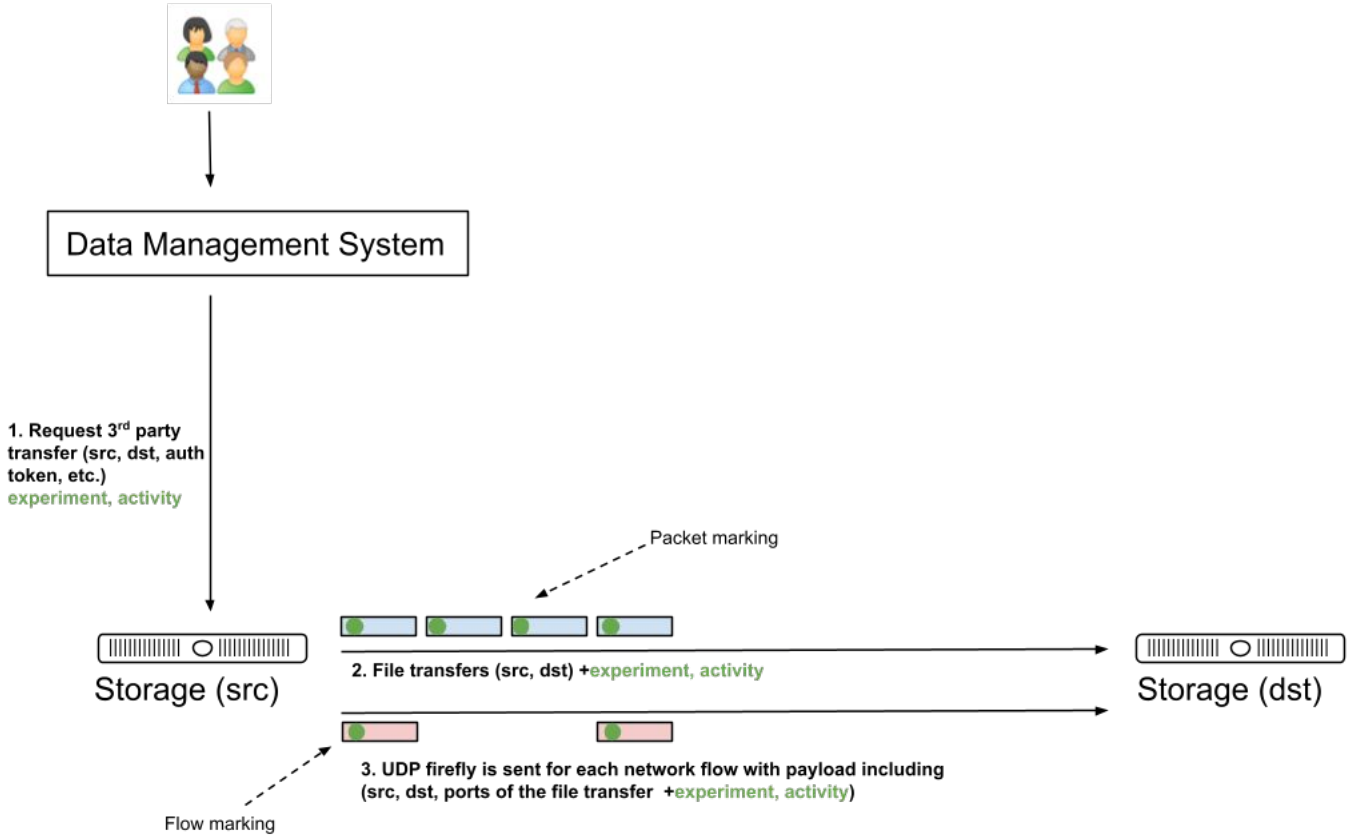
- Networks are becoming more programmable and capable with technologies such as P4, SDN, virtualisation, eBPF, etc.
- But with less and less context about the traffic they carry.
  - Cloud deployments, Kubernetes, encryption, tunneling, privacy, etc.
- Understanding scientific traffic flows in detail is critical for understanding how our complex systems are actually using the network.
  - Current monitoring/logging tell us where data flows start and end, but is unable to understand the data in flight.
  - Dedicated L3VPNs can be created to track high throughput science domains, but with more domains requiring high throughput this will become expensive, it won't scale, won't work at big sites having to support multiple domains at the same time.
- In general the monitoring we have is experiment specific and very difficult to correlate with what is happening in the network. We suggest this is a general problem for users of the Research and Education Networks (RENs).

Scitags Initiative

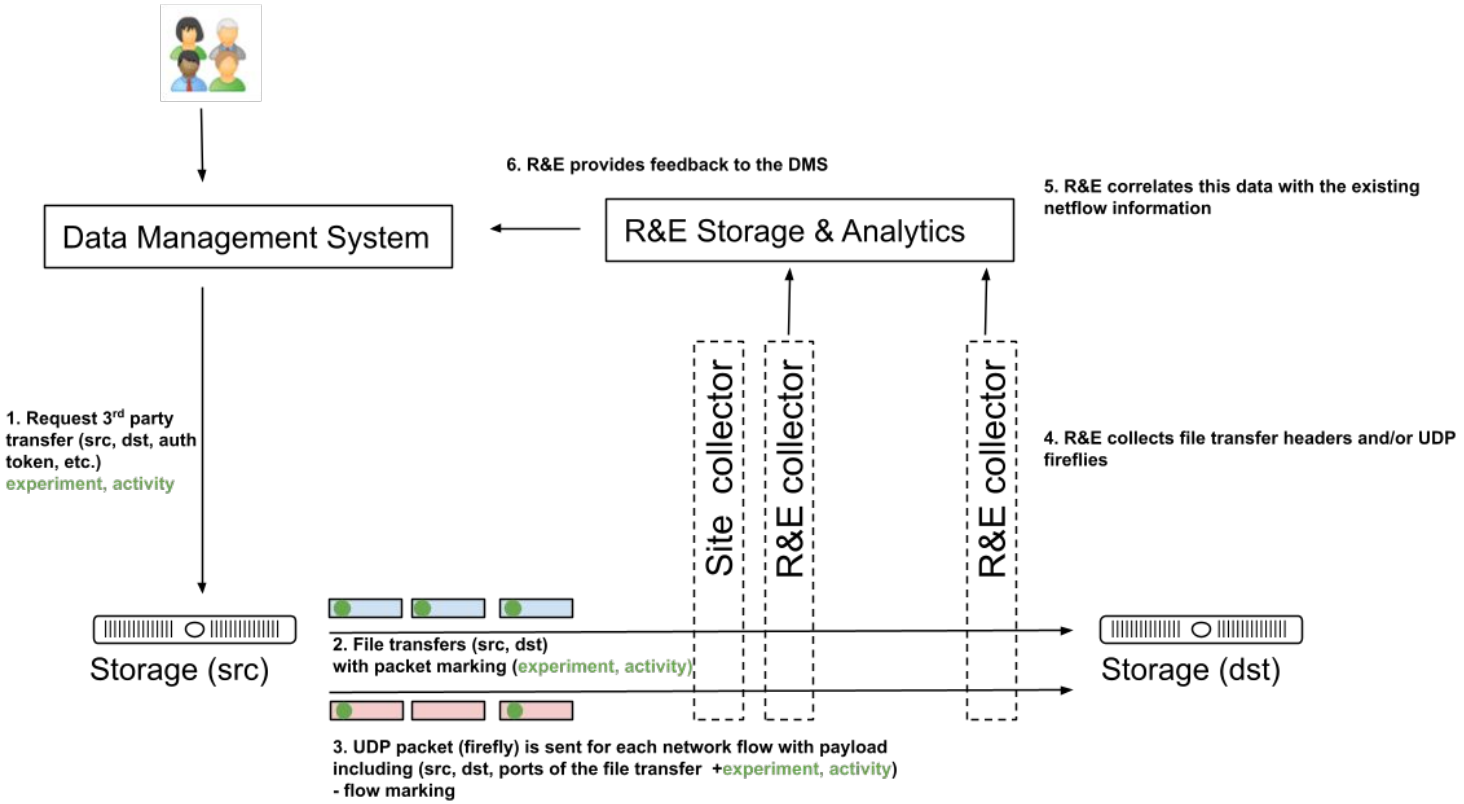
- Scientific Network Tags (scitags) is an initiative promoting identification of the science domains and their high-level activities at the network level.

- Enable tracking and correlation of our transfers with Research and Education Network Providers (R&Es) network flow monitoring
- Experiments can better understand how their network flows perform along the path
  - Improve visibility into how network flows perform (per activity) within R&E segments
  - Get insights into how experiment is using the networks, get additional data from R&Es on behaviour of our transfers (traffic, paths, etc.)
- Sites can get visibility into how different network flows perform
  - Network monitoring per flow (with experiment/activity information)
    - E.g. RTT, retransmits, segment size, congestion window, etc. all per flow

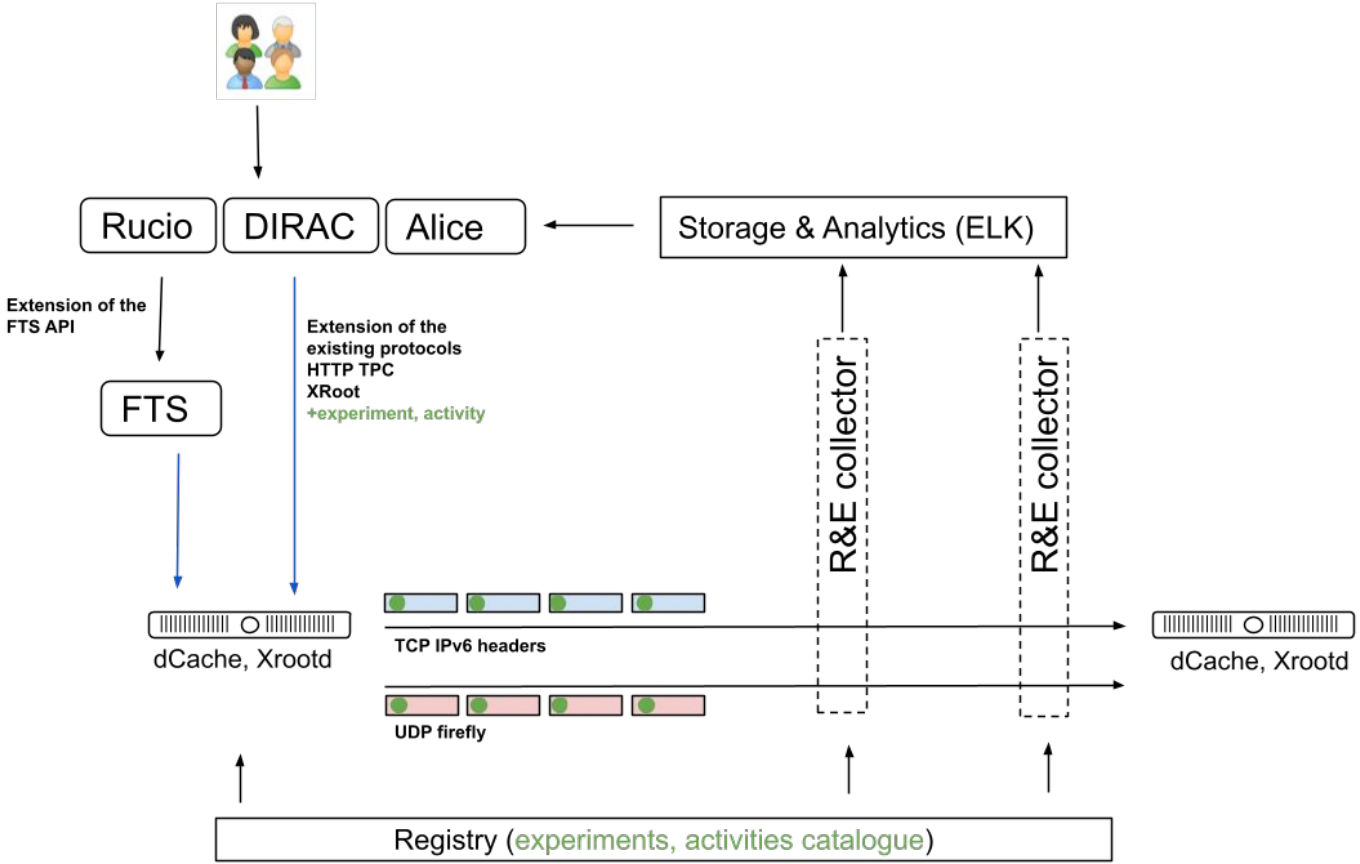
How scitags work



How scitags work



How scitags work



Technical Spec

The detailed technical specifications are maintained on a Google doc

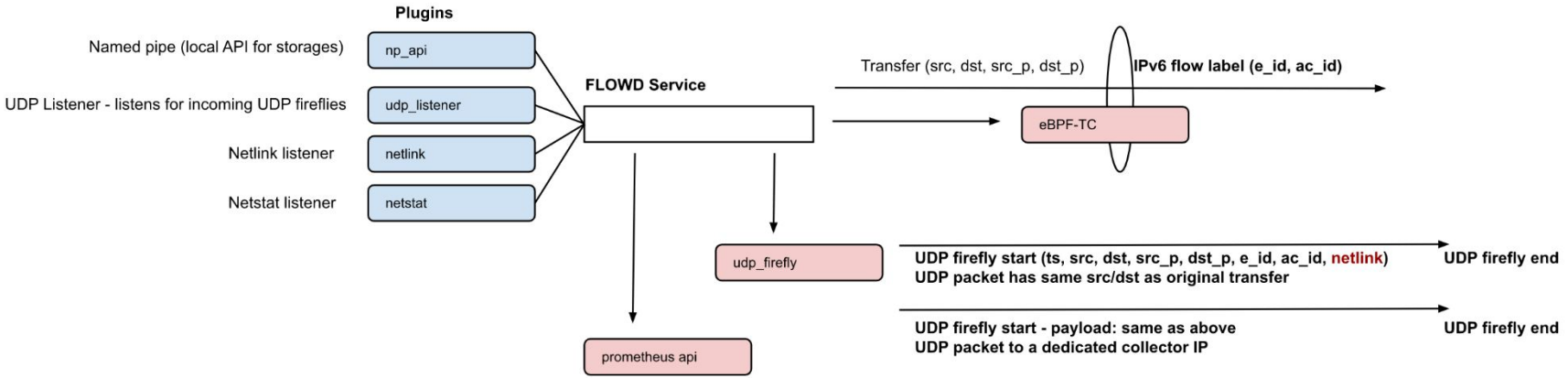
- The spec covers both Flow Labeling via UDP Fireflies and Packet Marking via the use of the IPv6 Flow Label.
  - **Fireflies** are UDP packets in Syslog format with a defined, versioned JSON schema.
    - Packets are intended to be sent to the same destination (port 10514) as the flow they are labeling and these packets are intended to be world readable.
    - Packets can also be sent to specific regional or global collectors.
    - Use of syslog format makes it easy to send to Logstash or similar receivers.
    - Two optional fields were added recently: **usage** and **netlink**
      - **Usage** reports on bytes sent/received as seen by the storage
      - **Netlink** adds TCP/IP low level metrics (RTT, Cwnd, Buffers, Busy time, etc.)
  - **Packet marking** uses the 20 bit flow label field in IPv6 packets.
    - To meet the spirit of RFC6437, we use 5 of the bits for entropy, 6 bits (64) for activity and 9 bits (512) for science domain (experiment).
- The document also covers methods for communicating owner/activity and other services and frameworks that may be needed for implementation.

Status

- **Flow Marking** (UDP firefly) implementations
  - **Xrootd** 5.0+ supports UDP fireflies
    - Site admins can configure mapping of paths to experiments and user/roles to activities
    - Xroot protocol extension via **scitag** URL flag
  - **dCache** PoC now ready, supports UDP fireflies
    - Test deployment at AGLT2 (backported to 7.2)
- **Flow and Packet Marking**
  - **Flowd** - packet and flow marking service
    - Independent service that can mark flows and packets for 3rd party services
- **Collectors/Receivers**
  - Initial receiver prototype developed by ESnet (available on scitags github)
- **Registry**
  - Provides list of experiments and activities supported
  - Exposed via JSON at api.scitags.org
- **Flow id propagation**
  - Work needed has been agreed with Rucio and FTS (tickets were submitted to follow up)
- Simplified deployment was tested during the last DC (& still operating)
- SC22 Demo

Flowd: News

- Flow and Packet Marking service developed in Python
  - Aims to provide a framework to test/evaluate various flow/packet marking ideas
  - Architecture:



- Plugins provide different ways get connections to mark (or interact with storage)
  - New plugins were added to support netlink readout and UDP firefly consumer
- Backends are used to implement flow and/or packet marking
  - New backends were added to mark packets (via eBPF-TC) and expose monitored connection to Prometheus

Flowd: eBPF-TC Backend

- eBPF is a general-purpose RISC instruction set that runs on an in-kernel VM; programs can be written in restricted C and compiled into bytecode that is injected into the kernel (after verification)
- Can sometimes replace kernel modules
- eBPF-TC programs run whenever the kernel receives (ingress hook) or sends (egress hook) a packet

- The flowd backend maintains a hash table of flows to mark. The plugin sends the backend (src address, dst address, src port, dst port); this is used as the key in the hash, and the flow label to put on the packets is the value
- Each packet is inspected, and if the attributes match an entry in the hash, the corresponding flow label is put on the packet

Flowd: Prometheus Client Backend

- Flowd keeps internal cache of all connections that it needs to track
- The plugin queries netlink to get snapshot of internal TCP/IP information per connection and exposes this information via Prometheus client
  - Netlink Information includes over 40 metrics that provide insight into TCP/IP kernel stack status
    - Examples include buffers (skmem info), congestion algorithm details (Cwnd, multipliers, BBR details, pacing/delivery rates, etc), Bytes/segments sent/received, MSS (thresholds, current values)

Traffic Shaping and Network Orchestration

The RNTWG has (so far) focused on the network visibility area due to limited manpower, but we have two additional areas the are part of the overall group goal: traffic shaping and network orchestration

Traffic Shaping:

- The WLCG experiments would like to explore traffic shaping/packet pacing.
  - Without packet pacing, network packets are emitted by the network interface in bursts, corresponding to the wire speed of the interface.
    - **Problem**: microbursts of packets can cause buffer overflows
    - The impact on TCP throughput, especially for high-bandwidth transfers on long network paths can be **significant**.
- Instead, pacing flows to match expectations [min(SRC,DEST,NET)] smooths flows and significantly reduces the microburst problem.
  - Broad implementation of pacing could make it feasible to run networks at much higher occupancy before requiring additional bandwidth
  - Existing R&D in eBPF-TC could be beneficial in this area as it can be reused also for packet pacing

Network Orchestration:

- This effort is being led by the GNA-g and includes work from the SENSE and FABRIC projects.
- RNTWG Session on the P4/programmable networks in September
  - Introduced R&D that would make it possible to perform **routing and forwarding based on IPv6 flow label**
  - This way we could separate traffic for different science domains (multiONE)

Network Visibility Plans

- Near-term objectives
  - Start rollout and testing of Xrootd implementation
    - Detect flow identifiers from storage path/url, activities from user role mapping
    - Test proxies, cached proxies, private networks (K8s)
  - Finalise development and deploy network of receivers
  - Instrument Rucio/FTS to pass flow identifiers to the storages
  - Involve other storage systems (dCache, etc.); discuss possible design/implementation

- Engage other R&Es and explore available technologies for collectors
  - Deploy additional collectors and perform R&D in the packet collectors
  - Improve existing data collection and analytics
- Test and validate ways to propagate flow identifiers
  - Engage experiments and data management systems
  - Validate, test protocol extensions and FTS integration
  - Explore other possibilities for flow identifier propagation, e.g. tokens
- Plan to submit IETF Informational RFC

Summary

The RNTWG has made significant progress on network traffic visibility through the work on flow labeling and packet marking.

- There remains a significant amount of work to do, especially regarding enabling packet marking on our storage infrastructure and in the area of collecting, aggregating and making visible the marked traffic.

We have additional work to pursue in traffic shaping:

- While network orchestration has significant activity underway, we need to find new effort interested in developing, prototyping and evaluating traffic shaping

We are always looking for additional manpower to join the effort!

Finding More Information: https://scitags.org

Code

Technical Spec

Mailing List

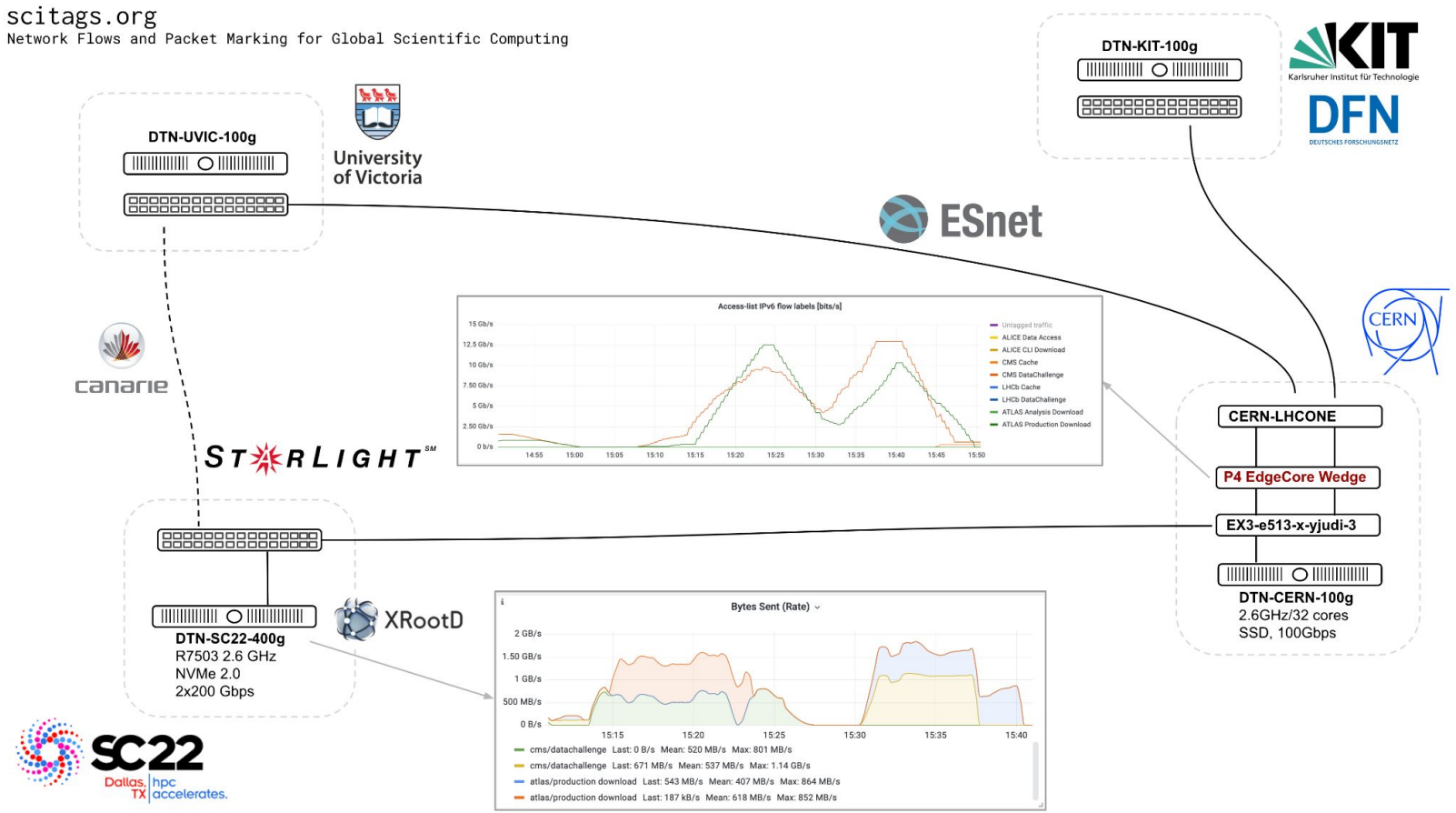
Presentations

SC22 Packet and Flow Marking Demo

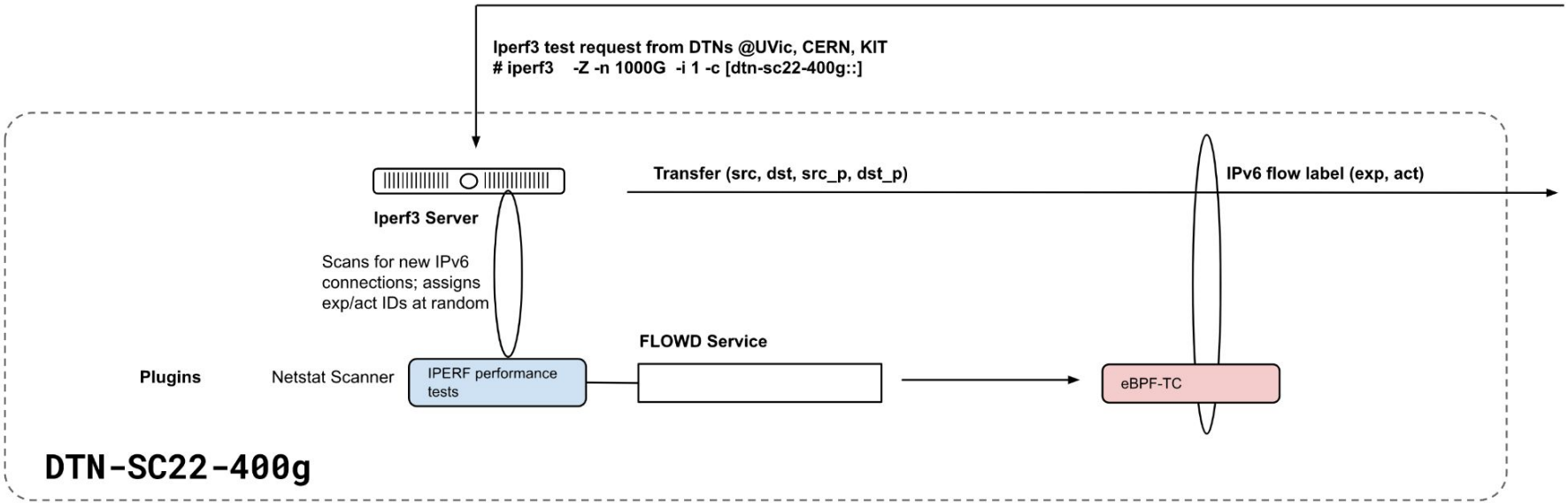
SC22 NRE

- The SC22 goals of the Packet and Flow Marking NRE demonstration will be to showcase the capabilities of the Scitags architecture and methods for optimizing data intensive science.
- Three demos:
  - IPv6 packet marking with eBPF-TC (100Gbps)
  - XRootD flow and packet marking with flowd + eBPF-TC
  - Accounting of flow label tagged packets using a P4 programmable switch
- Participants:
  - Starlight, CERN, University of Victoria, KIT, ESnet, Canarie

SC22 Demo


scitags.org
Network Flows and Packet Marking for Global Scientific Computing
DTN-KIT-100g
KIT
Karlsruher Institut für Technologie
DFN
DEUTSCHES FORSCHUNGSNETZ
DTN-UVIC-100g
University of Victoria
ESnet
CERN
canarie
STARLIGHT
Access-list IPv6 flow labels [bits/s]
15 Gb/s
12.5 Gb/s
10 Gb/s
7.50 Gb/s
5 Gb/s
2.50 Gb/s
0 b/s
14:55
15:00
15:05
15:10
15:15
15:20
15:25
15:30
15:35
15:40
15:45
15:50
Untagged traffic
ALICE Data Access
ALICE CLI Download
CMS Cache
CMS DataChallenge
LHCb Cache
LHCb DataChallenge
ATLAS Analysis Download
ATLAS Production Download
CERN-LHCONE
P4 EdgeCore Wedge
EX3-e513-x-yjudi-3
DTN-CERN-100g
2.6GHz/32 cores
SSD, 100Gbps
XRootD
DTN-SC22-400g
R7503 2.6 GHz
NVMe 2.0
2x200 Gbps
Bytes Sent (Rate)
2 GB/s
1.50 GB/s
1 GB/s
500 MB/s
0 B/s
15:15
15:20
15:25
15:30
15:35
15:40
cms/datachallenge Last: 0 B/s Mean: 520 MB/s Max: 801 MB/s
cms/datachallenge Last: 671 MB/s Mean: 537 MB/s Max: 1.14 GB/s
atlas/production download Last: 543 MB/s Mean: 407 MB/s Max: 864 MB/s
atlas/production download Last: 187 kB/s Mean: 618 MB/s Max: 852 MB/s
SC22
Dallas, TX
hpc accelerates.

IPv6 Packet Marking with eBPF-TC and P4 accounting



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

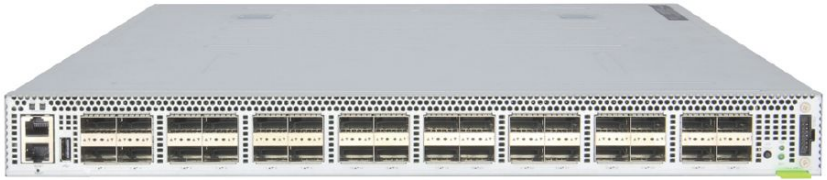
- ❑ Intel Tofino Switch Silicon
- ❑ Barefoot Networks

Quad-Pipe Programmable Packet Processing Pipeline

- ❑ 6.4 Tbps Total Bandwidth

CPU: Intelx86 Xeon 2.0GHz

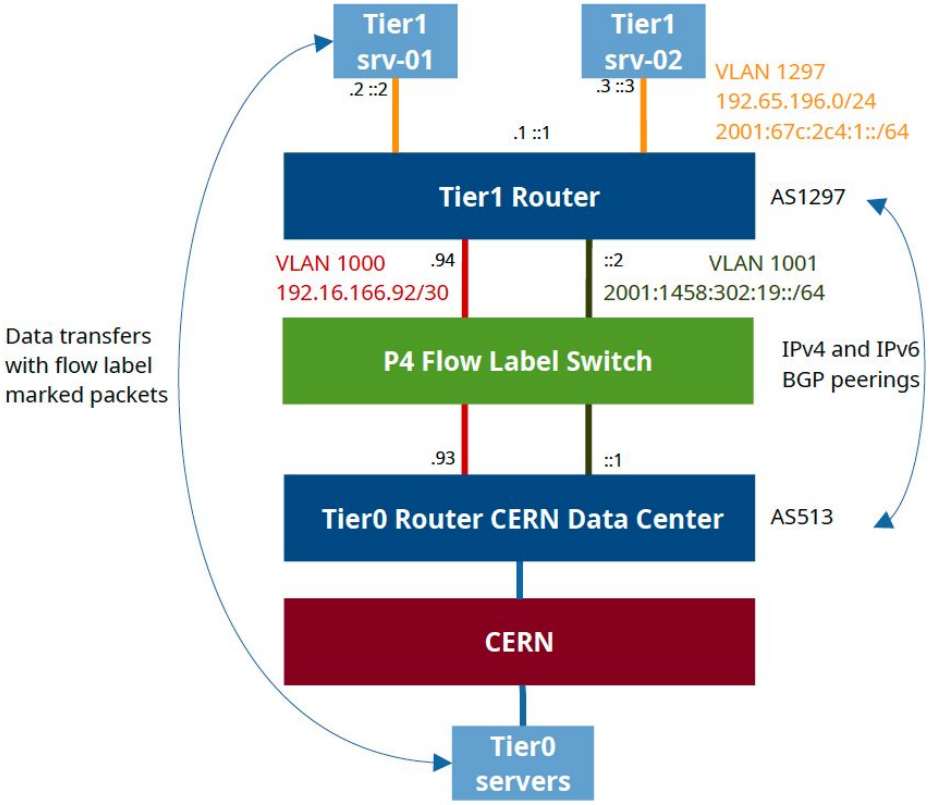
- ❑ 8-core/48GB/2TB SSD



Test: second approach at Ethernet level

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 product servers



P4 switch network configuration

Pure layer 2 bridges:

```
access-list acl_all_ipv6_flowlabels
     # Match <Experiment> and <ANY Application>
     sequence 10 permit all any all any all flow 131076 & 261884
     sequence 11 permit all any all any all flow 65540  & 261884
     sequence 12 permit all any all any all flow 196612 & 261884
     sequence 13 permit all any all any all flow 32772  & 261884
     # 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 196608 & 261632
     sequence 23 permit all any all any all flow 32768  & 261632
     # Permit the rest of the traffic
     sequence 30 permit all any all any all
     exit
```

```
ATLAS <any>
CMS   <any>
LHCb  <any>
ALICE <any>
```

```
ATLAS <perfSONAR>
CMS   <perfSONAR>
LHCb  <perfSONAR>
ALICE <perfSONAR>
```

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

VLAN 1000 belongs to bridge 1

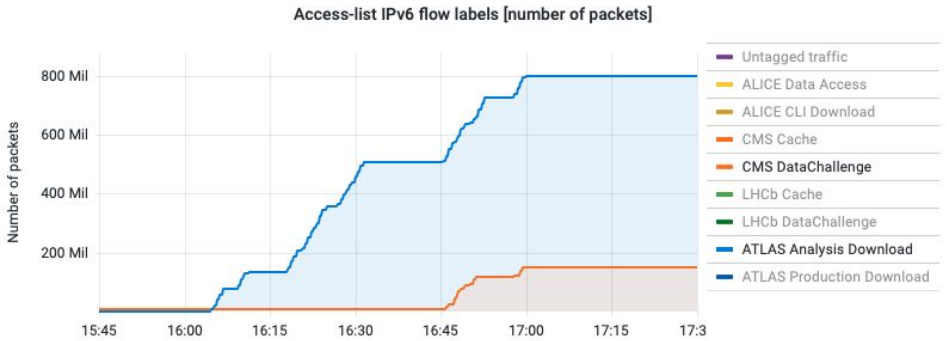
```
interface sdn1.1001
     description [VLAN ID=1001]
     bridge-group 2
     bridge-filter ipv6in acl_all_ipv6_flowlabels
     no shutdown
     no log-link-change
     exit
```

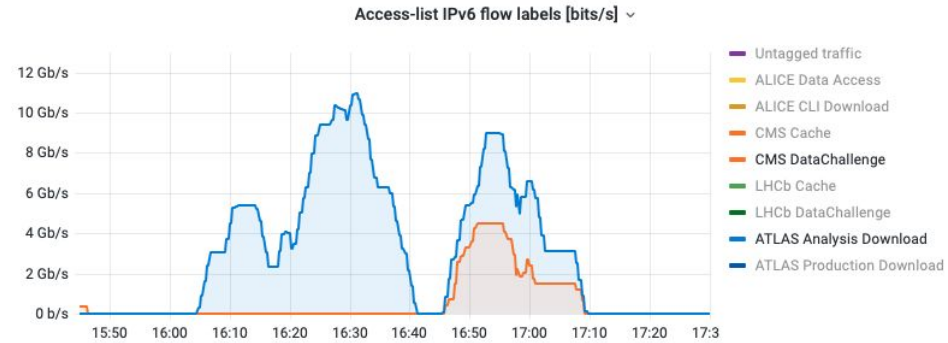
VLAN 1001 belongs to bridge 2
Filter IPv6 traffic at the input based on the access-list sentences

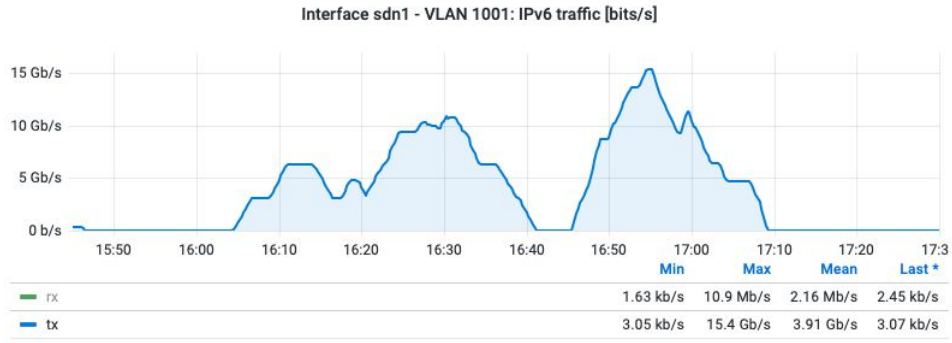
Statistics

The counters of the access-list of the P4 switch are exported to Prometheus DB and Grafana:

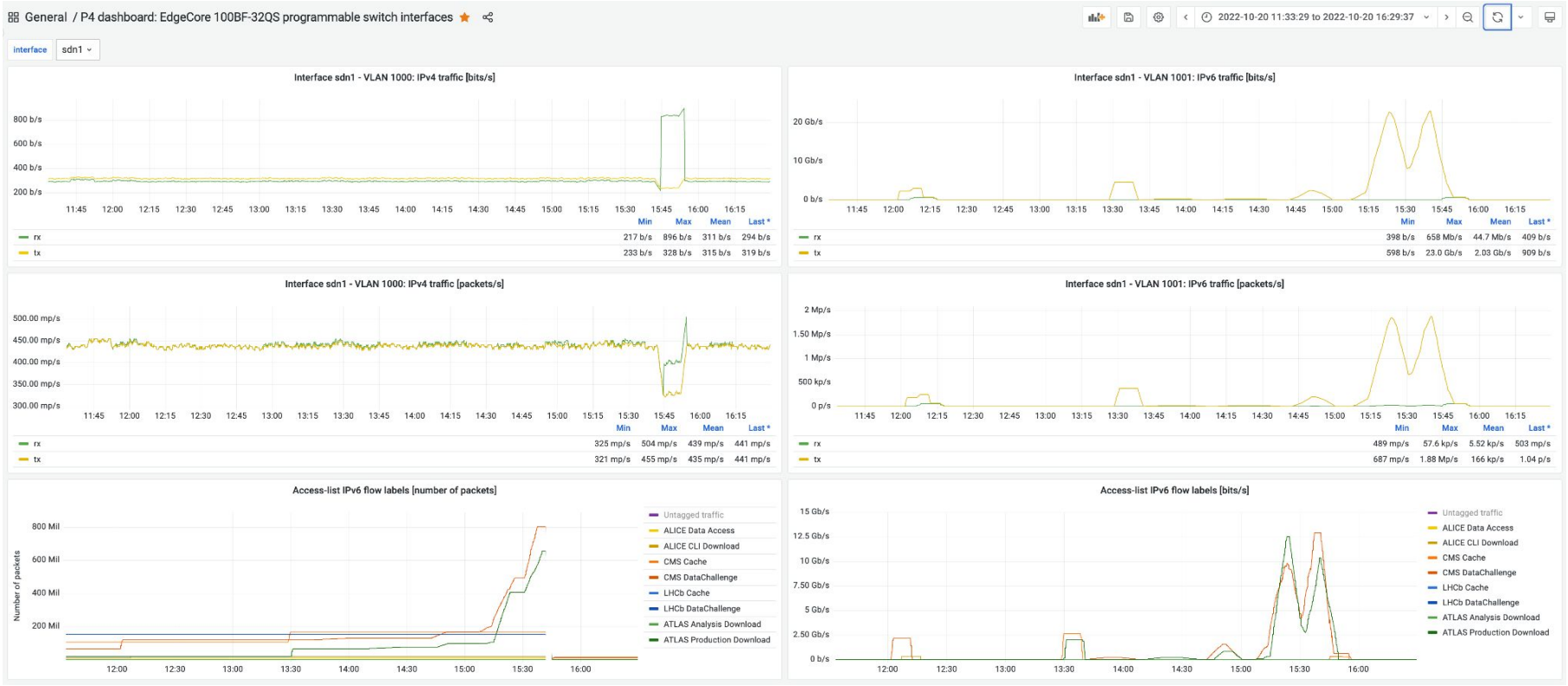
```
E513-E-YECWH-1#show access-list acl_all_ipv6_flowlabels
seq  txb  txp  rxb            rxp          last      timout    cfg
10   0+0  0+0  0+12374638771  0+8743031    00:03:02  00:00:00  permit all any all any all flow 131076&261884   ATLAS <any>
11   0+0  0+0  0+37019728635  0+24984028   00:02:30  00:00:00  permit all any all any all flow 65540&261884    CMS   <any>
12   0+0  0+0  0+23940164205  0+15797973   00:02:00  00:00:00  permit all any all any all flow 196612&261884   LHCb  <any>
13   0+0  0+0  0+18150017192  0+12017039   00:02:00  00:00:00  permit all any all any all flow 32772&261884    ALICE <any>

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 196608&261632   LHCb  <perfSONAR>
23   0+0  0+0  0+37078713993  0+25770785   00:00:26  00:00:00  permit all any all any all flow 32768&261632    ALICE <perfSONAR>
30   0+0  0+0  0+2715536713   0+1802921    00:00:26  00:00:00  permit all any all any all
```



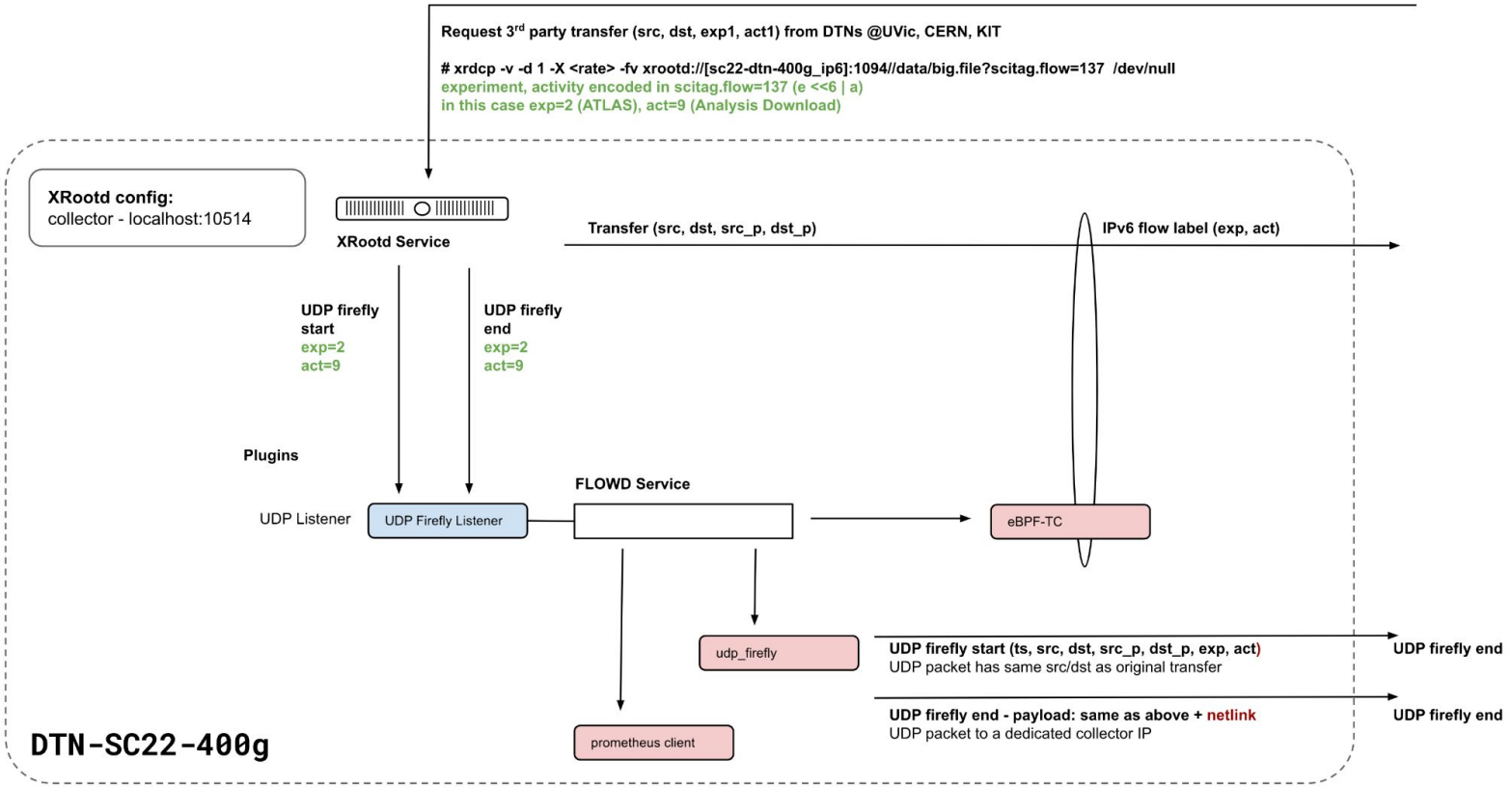




P4 Dashboard



XRootD Packet and Flow Marking and Netlink



Netlink Dashboard

Questions or Comments

Happy to take any questions, comments or suggestions!

Acknowledgements

We would like to thank the **WLCG**, **HEPiX**, **perfSONAR, CERN** and **OSG** organizations for their work on the topics presented.

In addition we want to explicitly acknowledge the support of the **National Science Foundation** which supported this work via:

- OSG: NSF MPS-1148698
- IRIS-HEP: NSF OAC-1836650

Backup slides

Registry

We need to standardize the “experiment” and “activity” fields we use for both flow labeling and packet marking.

The scitags.org domain provides an API that can be consulted to get the standard values: https://api.scitags.org or https://www.scitags.org/api.json

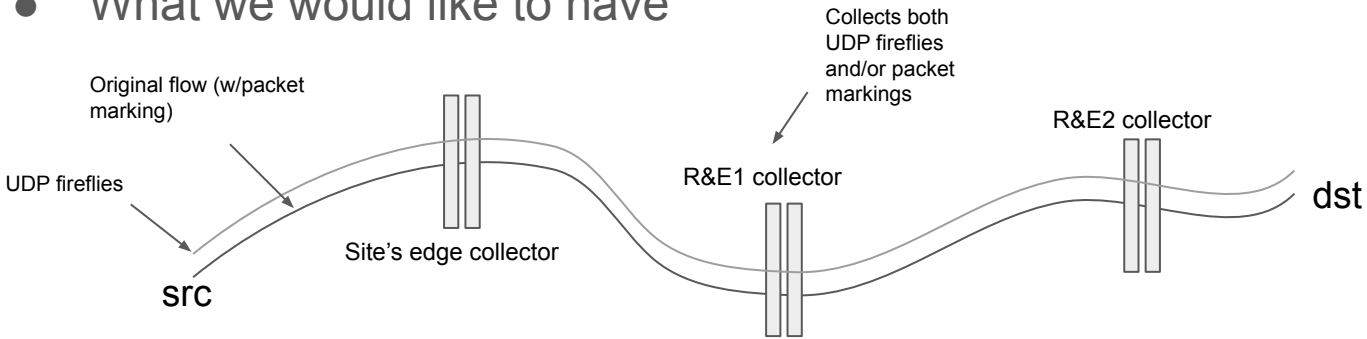
The underlying source of truth is a set of Google sheets that are maintained and writeable by a few stewards.

Note: the API provides the defined values **but** how the values are used in packet marking are specified in our Google sheets (bit location in IPv6 flow label)


```
{
  - experiments: [
      - {
            expName: "default",
            expId: 1,
          - activities: [
              - {
                    activityName: "default",
                    activityId: 1
                }
            ]
        },
      - {
            expName: "atlas",
            expId: 2,
          - activities: [
              - {
                    activityName: "perfsonar",
                    activityId: 2
                },
              - {
                    activityName: "cache",
                    activityId: 3
                },
              - {
                    activityName: "datachallenge",
                    activityId: 4
                },
              - {
                    activityName: "default",
                    activityId: 8
                },
              - {
                    activityName: "analysis download",
                    activityId: 9
                },
              - {
                    activityName: "analysis download direct io",
                    activityId: 10
```

Collectors

- What we would like to have

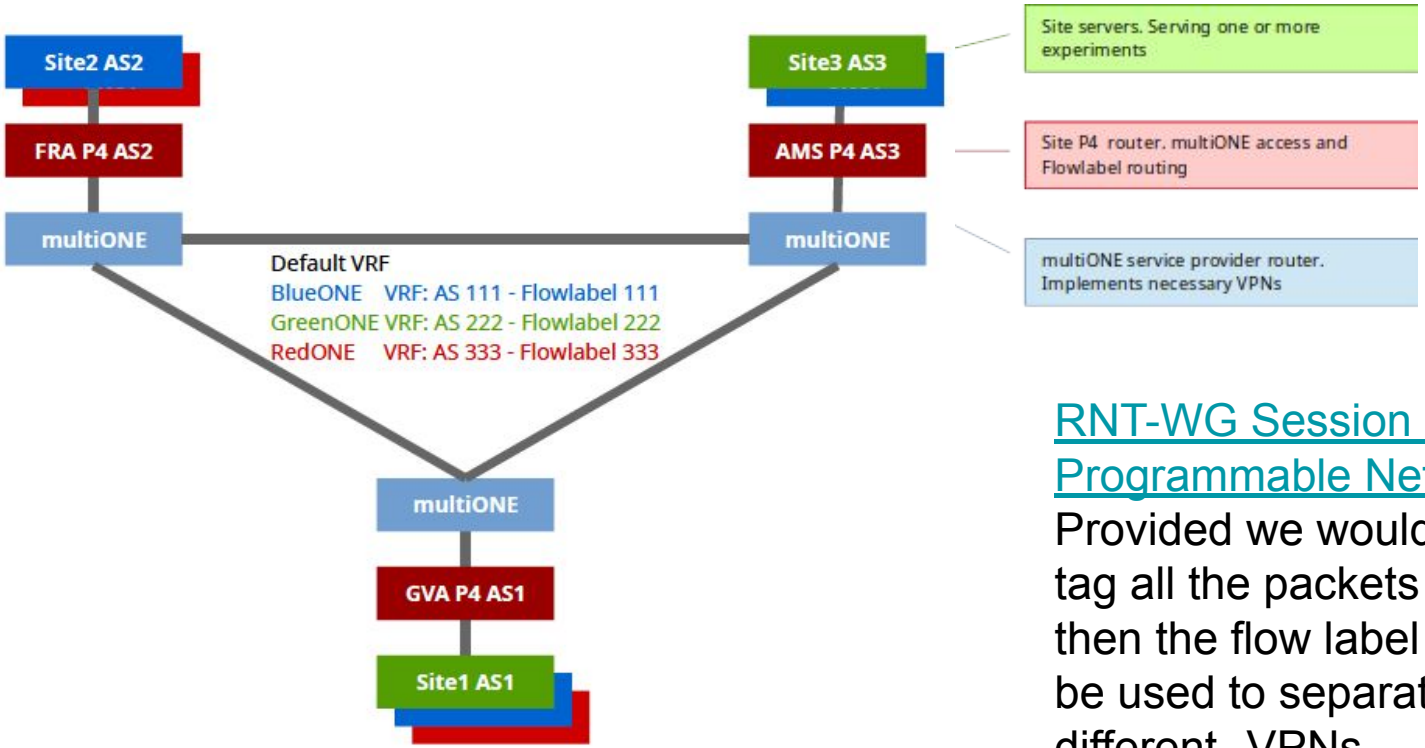


- Enable collection of packet and flow markings along the path
  - In order to extend R&E netflow information with flow identifier (experiment + activity)
  - UDP firefly packets needs to be collected and relayed to ensure they reach all collectors
- Each R&E **can** setup and operate one or more collectors
- Sites have an option to set up their own collector at the edge

Collectors

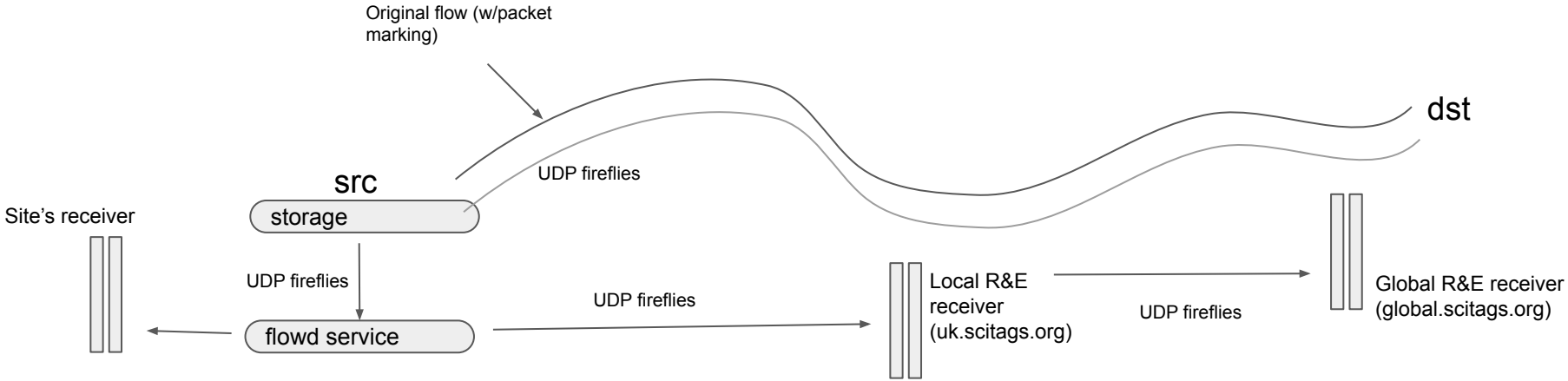
- Our **recommendation** is to use hardware/in-line collectors where possible
  - Requires port mirroring or other means to capture the fireflies.
  - Easiest to organise and operate as there is no need for a separate collector network.
  - Only way to capture flow markings along the path.
- However, in-line collectors require the ability to either selectively identify and capture fireflies or the ability to capture IPv6 flow labels from packets
  - Many possible ways to implement.
  - Strategy and technology to implement will depend on the R&E, their topology and hardware.
  - Would be great to get example implementations that can be shared between R&E network operators.

P4 Routing and Forwarding with IPv6 flow label



RNT-WG Session on P4/ Programmable Networks: Provided we would be able to tag all the packets in a flow, then the flow label tags could be used to separate traffic in different VPNs

Network of receivers



- Storages are configured with predefined DNS aliases (based on region; hosted by scitags.org)
  - Flowd service will expose API for site's local receivers and will also forward UDP fireflies to R&E collector (storage will send fireflies along the path)
  - Local R&E collector can be established (optional) and will need to pass all received fireflies to the global one (can switch to TCP)
- Works with inline/hardware collectors (which can be setup in parallel)
- Easy way to setup local R&E receiver (and correlate with local netflow)
- Lightweight - should be easy to operate, but requires some development in flowd and in the R&E collector
- DNS aliases will give us flexibility to make changes in the future (e.g. move to anycast)

WLCG Network Requirements

- Many WLCG facilities need network equipment refresh
  - Current routers in some sites are End-Of-Life and moving out of warranty
  - Local area networking often has 10+ year old switches which are no longer suitable for new nodes or operating at our current or planned scale.
- WLCG planning is including networking to a much greater degree than before
  - HL-LHC computing review: DOMA, dedicated networking section
  - ATLAS HL-LHC Computing Conceptual Design Report, highlights needs
  - Both include input from HEPiX, LHCONE/LHCOPN and WLCG working groups
- **Requirements Summary**
  - **Capacity**: Run-3 moving to multiple 100G links for big sites, Run-4 targeting Tbps links
  - **Capability**: WLCG needs to understand the impact of new features in networking (SDN/NFV) by testing, prototyping and evaluating impact. They will need to evolve their applications, facilities and computing models to meet the HL-LHC challenges; *it will take time*.
  - **Visibility**: As the ESnet Blueprinting meetings have shown, our ability to understand our WAN network flows is too limited. We need new methods to mark and monitor our network use
  - **Testing**: We need to be able to develop, prototype and test network features at suitable scale

Packet Marking Challenges

We would like this to be applicable for ALL significant R&E network users/science domains, not just HEP

- Requires us to think broadly during design

How best to use the number of bits we can get?

- Need to standardize bits and publish and **maintain**!!
- Can we agree on some standard “type” bits?

What can we rely on from the Linux network stack and what do we need to provide?

What can the network operators provide for accounting?

Packet Marking - Storage Elements

The primary challenge here is in two areas:

1. Augmenting the existing storage system to be able to set the appropriate bits in the network packets
2. Communicating the appropriate bits as part of a transfer request
    a. Likely need some protocol extension to support this
    b. Other ideas?

Packet Marking - Jobs

As jobs source data onto the network OR pull data into the job, we should try to ensure the corresponding packets are marked appropriately

- Containers and VMs may allow this to be easily put in place
- Still need configuration options that specify the right bits
- Signalling to the “source” about what those bits are also needs to be in place

Packet Marking - IPv6

IPv6 incorporates a “Flow Label” in the header (20 bits)

Fixed header format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				Traffic Class								Flow Label																			
4	32	Payload Length																Next Header								Hop Limit							
8	64	Source Address																															
12	96																																
16	128																																
20	160																																
24	192	Destination Address																															
28	224																																
32	256																																
36	288																																

Packet Marking - IPv4

IPv4 incorporates a “Options” in the header (allowing to add more 32 bit words)

IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags				Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
24	192																																
28	224																																
32	256																																

Scitags Overview

Our goal is to “instrument” any research network flows such that they are identifiable at any location along the path.

Challenges:

- **IPv4** has no good location in the packet header for marking
- **IPv6** has the “Flow Label” but our use is somewhat in tension with the RFC
- Linux kernels prior to 5.x lack good support to modify the Flow Label
- The node transferring the data needs to mark or label but this may require modifying all transfer software
- How does information about owner and purpose get to the transfer node ?

To address these issues we have implemented a program of work which includes both “packet marking” (for IPv6) and “flow labeling” via “Fireflies” for IPv4 as well as a “flowd” service to encapsulate the needed capabilities via plugins