Research Networking Technical WG Update

Marian Babik, Shawn McKee

LHCOPN/LHCONE, March 2021

Research Networking Technical WG

- 95 members from ~ 50 organisations
  - Motivated by the WLCG Network requirements review (WG Charter)
- Making our network use visible (**marking**)
  - Understanding HEP traffic flows in detail is critical for understanding how our complex systems are actually using the network.
- Shaping WAN data flows (**pacing**)
  - It remains a challenge for HEP storage endpoints to utilize the network efficiently and fully. Mainly focused on pacing the flows to match link capacity and avoid microburst problem; but also looking into new congestion control algorithms.
- **Orchestrating** the network to enable multi-site infrastructures
  - Data Lakes, federated or distributed Kubernetes and multi-site resource orchestration will certainly benefit (or require) some level of WAN network orchestration to be effective.

Making our network use visible

Understanding HEP 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. **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 our RENs (Research and Education Networks)**

- The proposed work is to identify how we might label our traffic at the **packet level** to indicate which **experiment** and **activity** it is a part of.

- The technical work encompasses how to **mark traffic** at the network level, defining a standard set of markings, **provide the tools** to the experiments to make it easy for them to participate and define how the NRENs can **monitor/account** for such data.

Challenges

Aim at all significant R&E network users/science domains, not just HEP

- Required us to think broadly during design

How to distribute tags to applications ? How will governance work ?

Which IP fields we can use for marking ? How best to use the number of bits we can get?

- Need to **standardize** bits and **publish** and **maintain** !!

Can the bits easily consumed by network hardware / software?

What can the network operators provide for accounting?

What technologies can we use in the Linux network stack ??

Packet Marking

- Outline:
  - Packet Marking Technologies
  - Standardisation
  - Validation
  - Networking (R&E) Perspective
  - Applications Perspective
  - Experiments Perspective
- See Packet Marking Document for details

Packet Marking Technologies/Standards

Internet Protocols (IP)

- IPv4
  - Explored IPv4 options
- IPv6 labels/options
  - IPv6 header fields (Flow label, Traffic class) and IPv6 extensions (Hop-by-hop and Dst Options)
- IPv6 addressing
  - Using multiple IPv6 addresses on a host (use address to mark)

Multiprotocol Label Switching (MPLS)

Beyond Internet Protocols

- From Network Tokens talk: TLS extension, STUN packets (RFC5389)

Packet Marking/Trade-offs

- IPv4
  - Existing IPv4 options not a good fit
  - New/unknown options likely to be dropped
  - Agreed to focus on IPv6 for a first prototype and if possible backport the functionality
- IPv6 labels/options
  - Traffic class - quite limited (only 4 bits for local/exp), in addition can be set/reset by R&Es
  - IPv6 HbH and Dst Options
    - Concerns over reachability - would require extensive testing
    - Implementation quite complex (requires root and ancillary data)
  - Flow Label
    - Selected as the easiest to do for the first prototype (20 bits field in the header)
- IPv6 addressing
  - Clients would need to use multiple addresses (bind to source address)
  - Requires complex changes to the existing applications
- MPLS
  - Complex to implement (end-to-end mpls support needed) and/or inter-domain encapsulation

IPv6 Flow Label

RFCs (10 hits)		
RFC 1809 **Using the Flow Label Field in IPv6**	1995-06 6 pages	Informational RFC
RFC 3595 *(was draft-ietf-ops-ipv6-flowlabel)* **Textual Conventions for IPv6 Flow Label**	2003-09 6 pages	Proposed Standard RFC
RFC 3697 *(was draft-ietf-ipv6-flow-label)* **IPv6 Flow Label Specification**	2004-03 9 pages	Proposed Standard RFC Obsoleted by RFC6437
RFC 6294 *(was draft-hu-flow-label-cases)* **Survey of Proposed Use Cases for the IPv6 Flow Label**	2011-06 18 pages	Informational RFC
RFC 6436 *(was draft-ietf-6man-flow-update)* **Rationale for Update to the IPv6 Flow Label Specification**	2011-11 13 pages	Informational RFC
RFC 6437 *(was draft-ietf-6man-flow-3697bis)* **IPv6 Flow Label Specification**	2011-11 15 pages	Proposed Standard RFC
RFC 6438 *(was draft-ietf-6man-flow-ecmp)* **Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels**	2011-11 9 pages	Proposed Standard RFC
RFC 7098 *(was draft-ietf-intarea-flow-label-balancing)* **Using the IPv6 Flow Label for Load Balancing in Server Farms**	2014-01 13 pages	Informational RFC

Document	Date	Status
Active Internet-Drafts (2 hits)		
draft-filsfils-6man-structured-flow-label-00 **Structured Flow Label**	2021-03-16 12 pages	I-D Exists New

Packet Marking Scheme

We started with **20 bits** (matching the size of the flow-label)

- We add 5 entropy bits to try to match the spirit of RFC6436
- We use 9 bits to define the Science Domain (reserving 3 for non-Astro/HEP domains)
- We use 6 bits to define the Application/Type of traffic
- We organize the bits to allow for potential adjustments in the future.

An initial packet marking scheme is in a Google sheet.

BitPattern	ScienceDomain	Application	Hdr Bit 12	Hdr Bit 13	Hdr Bit 14	Hdr Bit 15	Hdr Bit 16	Hdr Bit 17	Hdr Bit 18	Hdr Bit 23	Hdr Bit 24	Hdr Bit 29	Hdr Bit 30	Hdr Bit 31
xx100000000x000001xx	ATLAS	perfSONAR	x	x	1	0	0	0	0	x	0	1	x	x
xx010000000x000001xx	CMS	perfSONAR	x	x	0	1	0	0	0	x	0	1	x	x
xx110000000x000001xx	LHCb	perfSONAR	x	x	1	1	0	0	0	x	0	1	x	x
xx001000000x000001xx	ALICE	perfSONAR	x	x	0	0	1	0	0	x	0	1	x	x
xx101000000x000001xx	BelleII	perfSONAR	x	x	1	0	1	0	0	x	0	1	x	x
xx011000000x000001xx	SKA	perfSONAR	x	x	0	1	1	0	0	x	0	1	x	x
xx111000000x000001xx	LSST	perfSONAR	x	x	1	1	1	0	0	x	0	1	x	x
xx000100000x000001xx	DUNE	perfSONAR	x	x	0	0	0	1	0	x	0	1	x	x

Testing Flow Labels

- The first application used to test flow-label marking was perfSONAR Iperf3
  - We can centrally configure `--flow-label` for IPv6 **Iperf3** tests
  - Labels were manually verified via `tcpdump` at the destination
  - We now set a flow label on one perfSONAR test mesh (**US ATLAS**: CERN, AGLT2, MWT2, BNL, BU, LUNET, NERSC and Stanford; the flow label (65540) is set on iperf3 tests for this mesh.)

- IPv6 testing toolkit was enhanced by Fernando Gont to track flow labels
  - Developed as part of an effort to track IPv6 extension headers filtering
  - path6 - traceroute tool with full support of IPv6 extension headers
    - Following our request, the capability to track a particular flow label was introduced recently (howto)
  - Tim Chown has started an engagement with the **perfSONAR** developers to integrate the tool. This will provide an alternative to iperf3 which we can deploy

Flow Labels in Linux Kernel

- Ways to implement:
  - Advanced socket interface - native API
  - eBPF (XDP, BPF-TC) - run sandbox programs directly in Linux Kernel
  - DPDK, VPP - vendor-specific technologies
  - Software switches (via OpenFlow) and SmartNICs (via P4, etc.)

Examples using native socket API are now available, but come with limitations:

- Kernel 4.11+ (so C8+); Operational mode (label can only be set by the client, server only works if reflection is enabled); Optimisations (socket reuse ?), etc.
- Further tests are needed to better understand impact

Overall, quite complex situation in Linux as interface is currently split into two mutually exclusive use cases (load balancing vs flow manager facility)

Flow Labels readout

- IPFIX/Netflow
  - Standard approach on routers to sample traffic
  - IPFIX supports flow label, but this needs implementation in hardware/software:
    - Hardware: Nokia 7750 (non-encaps. only); Juniper (new chipset only); Cisco (IOS-XR 7.0.12+); Arista (IPFIX with flow labels in Q4 2020)
    - Software: nfcapd/nfdump, tshark, splunk support flow label
- SFlow
  - Unlike IPFIX significant part of the sampled raw packet is encapsulated and sent to the collector for analysis
    - Hardware: DELL EMC Networking OS9, OS10 (S6000-ON, S4248FB-ON); possibly supported also on other DC switches
    - Software: Elastiflow, sfcapd/nfdump

Current Plans and Schedule

- Continue to focus on **IPv6 Flow Label** option
- **Testing** in our R&E networks
  - Integration of path6 in perfSONAR to enable reachability testing
- **Applications** - enable packet marking in as many HEP applications as possible:
  - Improve existing examples for how to mark packets
    - Code examples for native Linux API now available
    - Working on initial code exercises with eBPF/XDP
    - Additional tests planned to understand coverage, limitations and support
  - We are currently targeting: perfSONAR, XRootD ASAP
    - We have initial xrootd plan, describing the work needed
    - Need broader engagement: FTS, Rucio, dCache, STORM, HTTP, etc.
- **Networks** - Consuming / Utilizing the bits
  - Work with R&E networks and sites to try to capture and measure the marked traffic
  - Verify traffic markings consistently pass end-to-end

Questions, Comments, Suggestions?

We have identified packet marking as important for WLCG and that work and the RNTWG parent group are a new focus area for the HEPiX working group.

We are interested in expanding membership.

We really need a broad range of expertise involved: network programming, standardization experience, experiment software expertise, storage software expertise, NRENs, documentation experience, monitoring, accounting, etc.

Questions, Comments, Suggestions?

Acknowledgements

We would like to thank the **WLCG**, **HEPiX**, **perfSONAR** 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

References

Packet marking document
Research Networking Technical WG Google folder
RNTWG Wiki
RNTWG mailing list signup
RNTWG/NFV WG Meetings and Notes: https://indico.cern.ch/category/10031/
NFV WG Report

Code examples

Research Networking Technical WG

Charter:
https://docs.google.com/document/d/1l4U5dpH556kCnoIHzyRpBl74IPc0gpgAG3VPUp98lo0/edit#

Mailing list:
http://cern.ch/simba3/SelfSubscription.aspx?groupName=net-wg

Members (90 as of today, in no particular order):

Christian Todorov (Internet2) Frank Burstein (BNL) Richard Carlson (DOE) Marcos Schwarz (RNP) Susanne Naegele Jackson (FAU) Alexander Germain (OHSU) Casey Russell (CANREN) Chris Robb (GlobalNOC/IU) Dale Carder (ESnet) Doug Southworth (IU) Eli Dart (ESNet) Eric Brown (VT) Evgeniy Kuznetsov (JINR) Ezra Kissel (ESnet) Fatema Bannat Wala (LBL) Joseph Breen (UTAH) James Blessing (Jisc) James Deaton (Great Plains Network) Jason Lomonaco (Internet2) Jerome Bernier (IN2P3) Jerry Sobieski Ji Li (BNL) Joel Mambretti (Northwestern) Karl Newell (Internet2) Li Wang (IHEP) Mariam Kiran (ESnet) Mark Lukasczyk (BNL) Matt Zekauskas (Internet2) Michal Hazlinsky (Cesnet) Mingshan Xia (IHEP) Paul Acosta (MIT) Paul Howell (Internet2) Paul Ruth (RENCI) Pieter de Boer (SURFnet) Roman Lapacz (PSNC) Sri N () Stefano Zani (CNAF) Tamer Nadeem (VCU) Tim Chown (Jisc) Tom Lehman (ESnet) Vincenzo Capone (GEANT) Wenji Wu (FNAL) Xi Yang (ESnet) Chin Guok (ESnet) Tony Cass (CERN) Eric Lancon (BNL) James Letts (UCSD) Harvey Newman (Caltech) Duncan Rand (Jisc) Edoardo Martelli (CERN) Shawn McKee (Univ. of Michigan) Simone Campana (CERN) Andrew Hanushevsky (SLAC) Marian Babik (CERN) James William Walder () Petr Vokac () Alexandr Zaytsev (BNL) Raul Cardoso Lopes () Mario Lassnig (CERN) Han-Wei Yen () Wei Yang (Stanford) Edward Karavakis (CERN) Tristan Suerink (Nikhef) Garhan Attebury (UNL) Pavlo Svirin () Shan Zeng (IHEP) Jin Kim (KISTI) Richard Cziva (ESnet) Phil Demar (FNAL) Justas Balcas (Caltech) Bruno Hoeft (FZK)

Review: 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 experiment's planning is including networking to a much greater degree than before
  - HL-LHC computing review: DOMA, dedicated networking section.
  - ESnet Planning and Case Studies: detailing operations, needs, use-case and future plans.
  - Broad realization that network challenges are going to be critical to prepare for HL-LHC
- **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 - IPv6

IPv6 candidates

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																																

Extension headers

For more details and discussion of various trade-offs please refer to the Packet Marking Document

Draft Packet Marking Scheme

We started with **20 bits** (matching the size of the flow-label)

- We add 5 entropy bits to try to match the spirit of RFC6436
- We use 9 bits to define the Science Domain (reserving 3 for non-Astro/HEP domains)
- We use 6 bits to define the Application/Type of traffic
- We organize the bits to allow for potential adjustments in the future.

The next few slides detail what we have arrived at

An initial packet marking scheme is in a Google sheet.

Science Domain Marking

The 9 bits assigned for Science Domain are in reverse bit-order

to keep the currently reserved (non-Astro/HEP) bits closest to the entropy bit, in case we need to adjust later. (Bits 11-9 != 0 are Non-Astro/HEP)

		LSB								MSB
DecimalValue	ScienceDomain	Hdr Bit 14	Hdr Bit 15	Hdr Bit 16	Hdr Bit 17	Hdr Bit 18	Hdr Bit 19	Hdr Bit 20	Hdr Bit 21	Hdr Bit 22
		Bit 17	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9
0	Reserved	0	0	0	0	0	0	0	0	0
65536	ATLAS	1	0	0	0	0	0	0	0	0
32768	CMS	0	1	0	0	0	0	0	0	0
98304	LHCb	1	1	0	0	0	0	0	0	0
16384	ALICE	0	0	1	0	0	0	0	0	0
81920	BelleII	1	0	1	0	0	0	0	0	0
49152	SKA	0	1	1	0	0	0	0	0	0
114688	LSST	1	1	1	0	0	0	0	0	0
73728	DUNE	1	0	0	1	0	0	0	0	0
8192		0	0	0	1	0	0	0	0	0

Application Marking Scheme

The 6 bits for Application are divided into two types: common across Science Domain (3 MSB = 0) and Science Domain specific

Note: some rows are hidden

We show the "**decimal value**" of the specific applications, assuming all the entropy bits are zero.

This makes it easy to add application+domain+entropy value to determine the final flow-label.

		MSB					LSB	
DecimalValue	Application	Hdr Bit 24	Hdr Bit 25	Hdr Bit 26	Hdr Bit 27	Hdr Bit 28	Hdr Bit 29	
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	
0	Reserved	0	0	0	0	0	0	Standardize for all Astro/HEP
4	perfSONAR	0	0	0	0	0	1	
8	Cache	0	0	0	0	1	0	
12		0	0	0	0	1	1	
16		0	0	0	1	0	0	
20		0	0	0	1	0	1	
24		0	0	0	1	1	0	
28		0	0	0	1	1	1	
32		0	0	1	0	0	0	Science Domain Specific
100		0	1	1	0	0	1	
104		0	1	1	0	1	0	
108		0	1	1	0	1	1	
112		0	1	1	1	0	0	
116		0	1	1	1	0	1	
120		0	1	1	1	1	0	
124		0	1	1	1	1	1	
128		1	0	0	0	0	0	
132		1	0	0	0	0	1	
136		1	0	0	0	1	0	
140		1	0	0	0	1	1	
144		1	0	0	1	0	0	
148		1	0	0	1	0	1	
152		1	0	0	1	1	0	
156		1	0	0	1	1	1	
160		1	0	1	0	0	0	
164		1	0	1	0	0	1	
168		1	0	1	0	1	0	
172		1	0	1	0	1	1	

Flow Labels in Linux

Our primary aim was to investigate Linux kernel native API. It allows fine-grained control over how flow labels are assigned and used. We have developed as set of examples how to enable, set and clear flow labels as well as how to read both local and remote labels.

Flow labels reflection is a feature that enables TCP server side support without any further implementation. UDP works as expected using extended recvfrom/sendto calls.

Currently only tested on the latest kernel version, further tests are needed to better understand support across different versions as well as impact of existing optimisations used in storage/transfer systems.

- Tests across WAN and other existing network middleware (proxies, etc.)
- Tests different kernel versions/distribution support (C8)

XDP/eBPF/BPF-TC

eBPF - technology that can run sandboxed programs in the Linux kernel - programs can be connected in the kernel space via various hooks (connection tracking, traffic control, etc.)

XDP uses eBPF for fast packet filtering and forwarding. It executes directly in the receive handler of the driver (driver hook/space; before packet gets to Linux network stack; before socket buffer metadata is created).

- **High-performance packet filtering, forwarding and manipulation (in place) is possible**
- **Can run as an independent service, tagging packets for all services**
  - Storage/transfer systems would only need to pass information on e.g. dst_addr/dst_port/tag
- **User space to kernel space communication can be used to configure the tagging**
  - Service can expose high level API to interact with
- **Easy integration with network namespaces, containers, etc.**

BPF-TC is another hook in traffic control (tc), which can work on egress but operates on socket buffers (TBD), which likely impacts its performance.