

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **SamplePacket API’s** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **02/05/2015** |
| **SAI-Version** | **V0.9.2** |

**Contents**

[List of Changes i](#_Toc415672730)

[1 Overview 1](#_Toc415672731)

[2 Specification 1](#_Toc415672732)

[2.1 Changes to sai.h 1](#_Toc415672733)

[2.2 Changes to saiport.h 1](#_Toc415672734)

[2.3 Changes to saiacl.h 1](#_Toc415672735)

[2.4 New definitions in saisamplepacket.h 2](#_Toc415672736)

[3 Example 4](#_Toc415672737)

[3.1 Create Samplepacket Session and get the session parameters 4](#_Toc415672738)

[3.2 Set Samplepacket attribute for the same session and verify the parameters using get session parameters. 5](#_Toc415672739)

[3.3 Enable/Assign the above samplepacket session to a port for ingress sampling 5](#_Toc415672740)

[3.4 Disable/Remove the above samplepacket session to a port for ingress sampling 6](#_Toc415672741)

[3.5 Enable/Assign the above samplepacket session to a port for egress sampling 6](#_Toc415672742)

[3.6 Disable/Remove the above samplepacket session to a port for egress sampling 6](#_Toc415672743)

[3.7 Enable samplepacket through acl (flow based sampling) 7](#_Toc415672744)

[3.8 Destroy samplepacket session 8](#_Toc415672745)

[4 Appendix 8](#_Toc415672746)

[**Subject:** RE: [Opencompute-networking] SAI - SFLOW comments/recommendations 8](#_Toc415672747)

[**Subject:** Re: [Opencompute-networking] SAI - SFLOW comments/recommendations 10](#_Toc415672748)

[**Subject:** RE: SAI - SFLOW comments/recommendations 13](#_Toc415672749)

[**Subject:** RE: SAI - SFLOW comments/recommendations 13](#_Toc415672750)

[**Subject:** RE: SAI - SFLOW comments/recommendations 14](#_Toc415672751)

[**Subject:** RE: SAI - SFLOW comments/recommendations 15](#_Toc415672752)

[**Subject:** RE: SAI - SFLOW comments/recommendations 15](#_Toc415672753)

[**Subject:** Re: [Opencompute-networking] SAI - SFLOW comments/recommendations 16](#_Toc415672754)

[**Subject:** [Opencompute-networking] SAI - SFLOW comments/recommendations 16](#_Toc415672755)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for sFlow – Version 1 |  | 2/5/15 |
| 0.9.2 | Version 2 Modified with the following changes   * Made SFLOW Enable/Disable a port based attribute * Removed the bit map definition for SFLOW Packet Type * Modified the comment for SFLOW Fast Path |  | 2/16/15 |
| 0.9.2 | Version 3 Removed the sFlow fast path attributes. Included in the appendix for future addition. |  | 3/11/15 |
| 0.9.2 | Version 4   * Removed the sFlow packet type attributes. Appendix section has the details. * Changed according to unified object id proposal |  | 3/17/15 |
| 0.9.2 | Version 5   * Added more details on the SAI\_SAMPLEPACKET\_ATTR\_SAMPLE\_RATE as per the discussions * Renamed sflow to sampleflow |  | 3/20/15 |
| 0.9.2 | Version 6   * Changed comments on the SAI\_SAMPLEPACKET\_ATTR\_SAMPLE\_RATE as per the discussions. * Renamed sampleflow to samplepacket * Changed the overview statement * Added Examples * Added couple of ACL Actions to sample packets matching an ACL |  | 3/31/15 |
| 0.9.2 | Version 7   * Removed ADMIN\_STATE attribute * Added details of how ACL based packet sampling should be done |  | 04/09/15 |

License

© 2014 Microsoft Corporation, Dell Inc., Facebook, Inc, Broadcom Corporation, Intel Corporation, Mellanox Technologies Ltd.

As of September 9, 2014, the following persons or entities have made this Specification available under the Open Web Foundation Final Specification Agreement (OWFa 1.0), which is available at <http://www.openwebfoundation.org/legal/the-owf-1-0-agreements/owfa-1-0>

Microsoft Corporation, Dell Inc., Facebook, Inc, Intel Corporation, Mellanox Technologies Ltd.

You can review the signed copies of the Open Web Foundation Agreement Version 1.0 for this Specification at <http://opencompute.org/licensing/>, which may also include additional parties to those listed above.

Your use of this Specification may be subject to other third party rights. THIS SPECIFICATION IS PROVIDED "AS IS." The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, noninfringement, fitness for a particular purpose, or title, related to the Specification. The entire risk as to implementing or otherwise using the Specification is assumed by the Specification implementer and user. IN NO EVENT WILL ANY PARTY BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE FOLLOWING IS A LIST OF MERELY REFERENCED TECHNOLOGY: Microprocessor technology, semiconductor manufacturing technology, operating system technology (including without limitation networking operating system technology), emulation technology, graphics technology, video technology, integrated circuit packaging technology and the like, compiler technologies, object oriented technology, optical/RF communications technology including chip I/O and driver technology, bus technology, memory chip technology (including, without limitation, NAND memory, NOR memory, resistive RAM (RRAM), seek scan probe (SSP) memory, nonvolatile memory (including without limitation, memory based on chalcogenide materials, phase change memory (PCM), one or more stacked layers of memory cells, embedded PCM memories, non-volatile cache memory, solid state drives, SRAM, embedded DRAM, ferro-electric memory, and polymer memory)) and/or health-related and medical technology. IMPLEMENTATION OF THESE TECHNOLOGIES MAY BE SUBJECT TO THEIR OWN LEGAL TERMS.

# Overview

The document describes covers details of the API to create, remove and update samplepacket object. The API allows user to create a SAI samplepacket object which defines the sampling rate, and then allows user to attach this sample object to various source SAI objects, such as port, ACL, etc. on.

Current proposal supports one sample destination which is the local CPU. In this type, all the sampled packets go to local CPU that is directly attached to the switching ASIC.

There are several ways to specify the sampling source. When attaching the SAI samplepacket object to a SAI port, all packets that go into this port or come out of this port are sampled. The proposal also support flow-based sampling which allows user to use ACL to specify the packets to sample.

# Specification

## Changes to sai.h

typedef enum \_sai\_api\_t {

SAI\_API\_SAMPLEPACKET= 15, /\* sai\_samplepacket\_api\_t \*

} sai\_api\_t;

## Changes to saiport.h

typedef enum \_sai\_port\_attr\_t

{

/\* READ-WRITE \*/

/\* Enable/Disable Samplepacket session [sai\_object\_id\_t]. (default to SAI\_NULL\_OBJECT\_ID)

\* Enable ingress sampling by assigning samplepacket object id as attribute value

\* Disable ingress sampling by assigning SAI\_NULL\_OBJECT\_ID as attribute value \*/ SAI\_PORT\_ATTR\_INGRESS\_SAMPLEPACKET\_SESSION\_ID

/\* READ-WRITE \*/

/\* Enable/Disable Samplepacket session [sai\_object\_id\_t].(default to SAI\_NULL\_OBJECT\_ID)

\* Enable egress sampling by assigning samplepacket object id as attribute value

\* Disable egress sampling by assigning SAI\_NULL\_OBJECT\_ID as attribute value \*/ SAI\_PORT\_ATTR\_EGRESS\_SAMPLEPACKET\_SESSION\_ID

} sai\_port\_attr\_t

## Changes to saiacl.h

typedef struct \_sai\_acl\_entry\_attr\_t {

/\* Ingress Samplepacket \*/

SAI\_ACL\_ENTRY\_ATTR\_ACTION\_INGRESS\_SAMPLEPACKET\_ENABLE,

/\* Egress Samplepacket \*/

SAI\_ACL\_ENTRY\_ATTR\_ACTION\_EGRESS\_SAMPLEPACKET\_ENABLE,

} sai\_acl\_entry\_attr\_t;

## New definitions in saisamplepacket.h

typedef enum \_sai\_samplepacket\_type\_t

{

/\* Copy the sample packets to CPU \*/

SAI\_SAMPLEPACKET\_SLOW\_PATH,

} **sai\_samplepacket\_type\_t**;

typedef enum \_sai\_samplepacket\_attr\_t

{

/\* READ-ONLY \*/

/\* READ-WRITE \*/

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET \*/

/\* Sampling rate of type uint32\_t. The sampling rate specifies random sampling probability as the ratio of packets observed to samples generated. For example a sampling rate of 100 specifies that, on average, 1 sample will be generated for every 100 packets observed. \*/ SAI\_SAMPLEPACKET\_ATTR\_SAMPLE\_RATE,

/\* CREATE\_ONLY \*/

/\* samplepacket switching type sai\_samplepacket\_type.

Default would be SAI\_SAMPLEPACKET\_SLOW\_PATH \*/

SAI\_SAMPLEPACKET\_ATTR\_TYPE,

} **sai\_samplepacket\_attr\_t**;

/\*

\* Routine Description:

\* Create samplepacket session.

\*

\* Arguments:

\* [out] session\_id - samplepacket session\_id

\* [in] attr\_count - number of attributes

\* [in] attr\_list - array of attributes

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_samplepacket\_session\_fn)(

\_Out\_ sai\_object\_id\_t \*session\_id,

\_In\_ uint32\_t attr\_count,

\_In\_ const sai\_attribute\_t \*attr\_list);

/\*

\* Routine Description:

\* Delete samplepacket session.

\*

\* Arguments:

\* [in] session\_id - samplepacket session\_id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_delete\_samplepacket\_session\_fn)(

\_In\_ sai\_object\_id\_t session\_id);

/\*

\* Routine Description:

\* Set samplepacket attributes on a port.

\*

\* Arguments:

\* [in] session\_id – session on which samplepacket should be set

\* [in] attr - attribute value

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_samplepacket\_attribute\_fn)(

\_In\_ sai\_object\_id\_t session\_id,

\_In\_ const sai\_attribute\_t \*attr);

/\*

\* Routine Description:

\* Get samplepacket attributes on a port.

\*

\* Arguments:

\* [in] session\_id - session on which samplepacket attributes should bre

\* [in] attr\_count - number of attributes

\* [inout] attr\_list - array of attributes

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_samplepacket\_attribute\_fn)(

\_In\_ sai\_object\_id\_t session\_id,

\_In\_ uint32\_t attr\_count,

\_Inout\_ sai\_attribute\_t \*attr\_list);

/\* SAMPLEPACKET method table retrieved with sai\_api\_query() \*/

typedef struct \_sai\_samplepacket\_api\_t

{

sai\_create\_samplepacket\_session\_fn create\_samplepacket\_session;

sai\_delete\_samplepacket\_session\_fn delete\_samplepacket\_session;

sai\_set\_samplepacket\_attribute\_fn set\_samplepacket\_attribute;

sai\_get\_samplepacket\_attribute\_fn get\_samplepacket\_attribute;

} **sai\_samplepacket\_api\_t**;

# Example

## Create Samplepacket Session and get the session parameters

sai\_api\_query(SAI\_API\_SAMPLEPACKET, &sai\_samplepacket\_api);

sai\_object\_id\_t sample\_object\_id;

sai\_attribute\_t attr[2];

attr[0].id = SAI\_SAMPLEPACKET\_ATTR\_SAMPLE\_RATE;

attr[0].value.u32 = 100;

attr[1].id = SAI\_SAMPLEPACKET\_ATTR\_ADMIN\_STATE;

attr[1].value.booldata = true ;

sai\_samplepacket\_api-> sai\_create\_samplepacket\_session\_fn(&sample\_object\_id, 2, attr);

sai\_attribute\_t get\_attr[2];

get\_attr[0].id = SAI\_SAMPLEPACKET\_ATTR\_SAMPLE\_RATE;

get\_attr[1].id = SAI\_SAMPLEPACKET\_ATTR\_ADMIN\_STATE;

sai\_samplepacket\_api-> sai\_get\_samplepacket\_attribute\_fn (sample\_object\_id, 2, get\_attr);

Check if get\_attr[0].value.u32 is 100

Check if get\_attr[1].value.booldata is true

## Set Samplepacket attribute for the same session and verify the parameters using get session parameters.

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_SAMPLEPACKET\_ATTR\_SAMPLE\_RATE;

set\_attr.value.u32 = 200;

sai\_samplepacket\_api-> sai\_set\_samplepacket\_attribute\_fn (sample\_object\_id, &set\_attr);

get\_attr.id = SAI\_SAMPLEPACKET\_ATTR\_SAMPLE\_RATE;

sai\_samplepacket\_api-> sai\_get\_samplepacket\_attribute\_fn (sample\_object\_id, 1, &get\_attr);

Check if get\_attr.value.u32 is 200

set\_attr.id = SAI\_SAMPLEPACKET\_ATTR\_ADMIN\_STATE;

set\_attr.value.booldata = false;

sai\_samplepacket\_api-> sai\_set\_samplepacket\_attribute\_fn (sample\_object\_id, & set\_attr);

get\_attr.id = SAI\_SAMPLEPACKET\_ATTR\_ADMIN\_STATE;

sai\_samplepacket\_api-> sai\_get\_samplepacket\_attribute\_fn (sample\_object\_id, 1, &get\_attr);

Check if get\_attr.value.booldata is false

## Enable/Assign the above samplepacket session to a port for ingress sampling

sai\_api\_query(SAI\_API\_PORT, &sai\_port\_api);

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_INGRESS\_SAMPLEPACKET\_ENABLE;

set\_attr.value.oid = sample\_object\_id;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, &set\_attr);

get\_attr.id = SAI\_PORT\_ATTR\_INGRESS\_SAMPLEPACKET\_ENABLE;

sai\_port\_api-> get\_port\_attribute(port\_object\_id, 1, &get\_attr);

Check if get\_attr.value.oid is sample\_object\_id

## Disable/Remove the above samplepacket session to a port for ingress sampling

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_INGRESS\_SAMPLEPACKET\_ENABLE;

set\_attr.value.oid = SAI\_NULL\_OBJECT\_ID;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, 1, &set\_attr);

## Enable/Assign the above samplepacket session to a port for egress sampling

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_SAMPLEPACKET\_ENABLE;

set\_attr.value.oid = sample\_object\_id;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, 1, &set\_attr);

get\_attr.id = SAI\_PORT\_ATTR\_INGRESS\_SAMPLEPACKET\_ENABLE;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, 1, &get\_attr);

Check if get\_attr.value.oid is sample\_object\_id

## Disable/Remove the above samplepacket session to a port for egress sampling

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr[1];

set\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_SAMPLEPACKET\_ENABLE;

set\_attr.value.oid = SAI\_NULL\_OBJECT\_ID;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, 1, &set\_attr);

## Enable samplepacket through acl (flow based sampling)

sai\_api\_query(SAI\_API\_ACL; &sai\_acl\_api);

sai\_acl\_table\_id\_t acl\_table\_id;

sai\_acl\_entry\_id\_t acl\_rule\_id;

sai\_attribute\_t attr[12] = {0};

attr[0].id = SAI\_ACL\_TABLE\_ATTR\_STAGE;

attr[0].value.s32= 0;

attr[1].id = SAI\_ACL\_TABLE\_ATTR\_PRIORITY;

attr[1].value.u32 = 1;

attr[2].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_SRC\_MAC;

attr[3].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_DST\_MAC;

attr[4].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_ETHER\_TYPE;

attr[5].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_IP\_TYPE;

attr[6].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_INNER\_VLAN\_ID;

attr[7].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_INNER\_VLAN\_PRI;

attr[8].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_INNER\_VLAN\_CFI;

attr[9].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_IP\_PROTOCOL;

attr[10].id = SAI\_ACL\_ENTRY\_ATTR\_FIELD\_IN\_PORT;

attr[11].id = SAI\_ACL\_ENTRY\_ATTR\_FIELD\_IN\_PORTS;

sai\_acl\_api->create\_acl\_table (&acl\_table\_id, 12, &attr);

sai\_attribute\_t rule\_attr[10] = {0};

rule\_attr[0].id = SAI\_ACL\_ENTRY\_ATTR\_TABLE\_ID;

rule\_attr[0].value.u32= acl\_table\_id;

rule\_attr[1].id = SAI\_ACL\_ENTRY\_ATTR\_PRIORITY;

rule\_attr[1].value.u32 = 1;

rule\_attr[2].id = SAI\_ACL\_ENTRY\_ATTR\_ADMIN\_STATE;

rule\_attr[2].value.booldata= true;

rule\_attr[3].id = SAI\_ACL\_ENTRY\_ATTR\_FIELD\_IN\_PORTS;

rule\_attr[3].value.portlist.port\_count = 1;

rule\_attr[3].value.portlist.port\_list = (sai\_port\_id\_t \*) calloc(

1, sizeof(sai\_port\_id\_t));

rule\_attr[3].value.portlist.port\_list[0] = 1;//Embed in sai\_object\_id port numbering format

rule\_attr[4].id = SAI\_ACL\_ENTRY\_ATTR\_ACTION\_ INGRESS\_ SAMPLEPACKET\_ENABLE;

sai\_acl\_api->create\_acl\_rule (&acl\_rule\_id, 5, &rule\_attr);

## Destroy samplepacket session

sai\_samplepacket\_api-> delete\_samplepacket\_session (sample\_object\_id);

# Appendix

**Subject:** RE: [Opencompute-networking] SAI - SFLOW comments/recommendations

Thanks Peter for your feedback

* Yes I will remove the packet type option so all packets are sampled
* Yes there would be additional meta data that would be passed from packet handler to above layers. This would be defined as part of the Host If Spec. Any other get functionality that agent needs to fill in it headers we could add later

On counters there is no cache at SAI layer. The SAI would return what the NPU SDK returns. Currently there is no switch attribute to define the counter refresh interval

-Arun

The following provides a list of possible attributes that can be added if an NPU supports sFlow in HW

typedef enum \_sai\_samplepacket\_attr\_t

{

…..

/\* All the attributes below are mandatory for SAI\_SAMPLEPACKET\_FAST\_PATH \*/

/\* CREATE + READ-ONLY \*/

/\* samplepacket Source MAC Address [sai\_mac\_t]\*/

SAI\_SAMPLEPACKET\_ATTR\_SRC\_MAC\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* samplepacket Destination MAC Address [sai\_mac\_t]\*/

SAI\_SAMPLEPACKET\_ATTR\_DST\_MAC\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* samplepacket VLAN Id [sai\_vlan\_id\_t]\*/

SAI\_SAMPLEPACKET\_ATTR\_VLAN\_ID,

/\* CREATE + READ-ONLY \*/

/\* samplepacket collector’s next hop information. Destination port shall be retrieved by that information [sai\_object\_id\_t]\*/

SAI\_SAMPLEPACKET\_ATTR\_DST\_PORT,

/\* CREATE + READ-ONLY \*/

/\* samplepacket collector address required for hardware switching [sai\_ip\_address\_t] \*/

SAI\_SAMPLEPACKET\_ATTR\_SRC\_IP\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* samplepacket agent address required for hardware switching [sai\_ip\_address\_t ] \*/

SAI\_SAMPLEPACKET\_ATTR\_DST\_IP\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* samplepacket Header UDP information [uint16\_t] \*/

SAI\_SAMPLEPACKET\_ATTR\_L4\_UDP\_DST\_PORT,

/\* CREATE + READ-ONLY \*/

/\* samplepacket Header UDP source port [uint16\_t] \*/

SAI\_SAMPLEPACKET\_ATTR\_L4\_UDP\_SRC\_PORT,

/\* CREATE + READ-ONLY \*/

/\* samplepacket Header Datagram size [uint16\_t] \*/

SAI\_SAMPLEPACKET\_ATTR\_L4\_UDP\_DATAGRAM\_SIZE,

} **sai\_samplepacket\_attr\_t**;

Thanks Peter for your feedback

* Yes I will remove the packet type option so all packets are sampled
* Yes there would be additional meta data that would be passed from packet handler to above layers. This would be defined as part of the Host If Spec. Any other get functionality that agent needs to fill in it headers we could add later

On counters there is no cache at SAI layer. The SAI would return what the NPU SDK returns. Currently there is no switch attribute to define the counter refresh interval

-Arun

**Subject:** Re: [Opencompute-networking] SAI - SFLOW comments/recommendations

Hi All,

I have started looking at the SAI documents and following the discussion and have a few comments about the sFlow extension.

Firstly, I think a standard API for sFlow agents to interact with hardware is a very positive step that will help ensure consistency and reduce the effort of implementing sFlow agents to different hardware platforms.

The sFlow version 5 specification (<http://sflow.org/sflow_version_5.txt>) doesn’t permit selective / filtered sampling of packets. The standard requires that a random sample of all packets transiting the data source be sampled with an equal probability.  Removing the \_sai\_sflow\_packet\_type\_t configuration options will help ensure that sFlow exports are compliant with the specification and consistent between implementations.

However, there are additional packet meta data attributes that should be passed up with the sampled packet in order to fill in the sFlow flow\_sample and sampled\_header structures:

/\* flow\_sample, page 29 \*/

sampling\_rate, i.e. the actual rate implemented in hardware which may differ from configured rate SAI\_SFLOW\_ATTR\_SAMPLE\_RATE (unless the semantics are defined to be consistent with sFlowFsPacketSamplingRate, page 20).

sample\_pool

drops

input ifIndex

output ifIndex

/\* sampled\_header, page 35 \*/

header\_protocol

frame\_length

stripped

header<>, up to sFlowFsMaximumHeaderSize bytes of the packet header + number of header bytes (must not include any padding)

Note: Typically the driver sends up the whole packet and the sFlow agent extracts the header and fills in the frame\_length, stripped and header<> fields.

The following attributes could be passed up with the metadata, or queried by the sFlow agent:

/\* extended\_switch, page 37 \*/

src\_vlan

src\_priority

dst\_vlan

dst\_priority

/\* extended\_router, page 38 \*/

nexthop

src\_mask\_len

dst\_mask\_len

/\* extended\_gateway \*/

nexthop

as

src\_as

src\_peer\_as

dst\_as\_path

communities

localpref

There are additional required fields such as sequence numbers, sub\_agent and data\_source values, but these are added by the software agent.

The counter export is an important part of sFlow export and it looks like the \_sai\_port\_stat\_counter\_t enum lists the basic port counters needed to populate the sFlow if\_counters and ethernet\_counters structures. The ifSpeed is in \_sai\_port\_attr\_t, but I couldn’t find the duplex mode.

I couldn’t find a discussion of when the counters are actually retrieved from the ASIC/NPU registers. The sFlow spec requires that the time between retrieving counters from hardware and sending them as an sFlow datagram be no more than 1 second. If there is no caching by the driver, then this requirement is easily met. If counters are cached by the driver, then a method of bypassing the cache, or a notification of updates, would allow the sFlow agent to satisfy this timing requirement.

There are other counters that are relevant to the functionality exposed by the SAI. The sFlow LAG extension (<http://sflow.org/sflow_lag.txt>) exports LACP counters and LAG containment relationships. The Host sFlow counters (<http://sflow.org/sflow_host.txt>) provide additional information that the sFlow agent can gather and export from the host O/S.

The Host sFlow (<http://host-sflow.sourceforge.net/>) provides an open source implementation of an sFlow agent that could easily be ported to Open Network Linux (the agent also provides an example of the interactions with the underlying operating system and drivers needed to support sFlow export).

There was a suggestion to use a generic name instead of sFlow to describe the APIs. However, sFlow is very specific about the method of packet sampling, counter polling, the form and meaning of exported data structures, and the timing of data export by the agent. Specificity is important since it ensures consistency and accuracy of sFlow implementations on different hardware platforms and between different device vendors. If the API definitions aren’t clear and don’t reference the underlying sFlow specification documents then it is difficult to ensure that sFlow implementations built on top of the APIs conform to the specifications and interoperate (the primary goal of the sFlow license - <http://sflow.org/developers/licensing.php> ).

Peter

On Mar 16, 2015, at 6:44 AM, Srini Addepalli <[saddepalli@freescale.com](mailto:saddepalli@freescale.com)> wrote:

I agree.  It is very difficult to get the filtering granularity right via SAI API. Since, it is not mentioned in the sflow MIB,  it is safe to remove and let host do any filtering they deem necessary before sending the samples to collectors.

Thanks  
Srini

**Subject:** RE: SAI - SFLOW comments/recommendations

Yes I am fine to remove it. Looks like some NPU’s support this, so let’s wait to hear for any feedback on this before we remove it. Any nonstandard thing can be part of vendor extensions

-Arun

**Subject:** RE: SAI - SFLOW comments/recommendations

Hi Arun,

Yes, and this is based on the RFC understanding where there is no such object in the MIB. The datagram sent to collector has this information and collector makes a picture of network state based on it. The sampling is independent of the type.

Regards,  
Atit

**Subject:** RE: SAI - SFLOW comments/recommendations

Are you suggesting any packet type should be monitored and not have a control of what packet type gets monitored or suggesting make this definition common

-Arun

**Subject:** RE: SAI - SFLOW comments/recommendations

Hi Arun,

Following is part of ver3 doc. My question was “(why) Also should we include the packet type(\_sai\_sflow\_packet\_type\_t) here”. I am also saying that the SAI api should not be aware. So we should remove following from the doc.

typedef enum \_sai\_sflow\_packet\_type\_t

{

    SAI\_SFLOW\_PACKET\_TYPE\_UNICAST ,

    SAI\_SFLOW\_PACKET\_TYPE\_MULTICAST,

    SAI\_SFLOW\_PACKET\_TYPE\_BROADCAST,

    SAI\_SFLOW\_PACKET\_TYPE\_UNKNOWN\_UNICAST ,

    SAI\_SFLOW\_PACKET\_TYPE\_UNREGISTERED\_MULTICAST ,

} **sai\_sflow\_packet\_type\_t**;

Regards,

Atit

**Subject:** RE: SAI - SFLOW comments/recommendations

Hi Atit,

My understanding is at SAI the SFLOW packet would be transparently passed up/down based on the RX/TX so why should SAI be aware of the SFLOW packet type being send. Are you asking for the case where SFLOW is supported in HW and we can define the SFLOW packet fields

On your other question, we have a mode set attribute which would return an ERROR is we try to set Fast path on a NPU that does not support switch handling of SFLOW samples.

-Arun

**Subject:** RE: SAI - SFLOW comments/recommendations

Hi Arun,

One comment inline. Also should we include the packet type(\_sai\_sflow\_packet\_type\_t) here, Ideally the packet related aspects are part of datagram sent to the collector. If we see the RFC (version 5) the MIB doesn’t have this attribute. Once enabled on the port/source, all the packets should be sampled. Let mi know if I missed something.

Regards,

Atit

I would expect the sai\_api\_query() to return NULL method table for SFLOW in this case

<Atit> the query only returns the function table (struct) pointer. I think Srini meant the Switch’s capability of sending samples/datagrams directly to the collector.

**Subject:** Re: [Opencompute-networking] SAI - SFLOW comments/recommendations

Thanks for the feedback Srini

Comments inline

**Subject:** [Opencompute-networking] SAI - SFLOW comments/recommendations

Hi,

Few comments :

1.       Counter sampling is equally important as packet sampling.  Please consider adding counter sampling capability.

The header file would get to OCP Git gub in the next few weeks. You can propose this on top of that

2.       Sflow packet and counter record transport is simple and many NPUs can send those directly to collectors without any intervention from the host.  Was there any particular reason to move this to appendix?  We would like to see this in the API.

We wanted to make the interface in case the agent is supported by NPU to be complete and cover all cases. I could not get hold of any implementation that fully supports this functionality in NPU to provide a good interface and hence deferred for subsequent update

3.       Since some switches don’t support transporting the packets samples and counter records,  it is good to have it exposed as switch capability.

I would expect the sai\_api\_query() to return NULL method table for SFLOW in this case

-Arun

Thanks  
Srini

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
opencompute-networking mailing list  
Unsubscribe: <http://lists.opencompute.org/mailman/options/opencompute-networking>  
  
[opencompute-networking@lists.opencompute.org](mailto:opencompute-networking@lists.opencompute.org)  
<http://lists.opencompute.org/mailman/listinfo/opencompute-networking>

sFlow Header Datagram size [uint16\_t] \*/

SAI\_SFLOW\_ATTR\_L4\_UDP\_DATAGRAM\_SIZE,

} sai\_sflow\_attr\_t