

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **QOS MAP API’s** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **03/03/2015** |
| **SAI-Version** | **V0.9.3** |

**Contents**

[List of Changes i](#_Toc417378492)

[1 Overview 1](#_Toc417378493)

[2 Specification 1](#_Toc417378494)

[2.1 Changes to sai.h 1](#_Toc417378495)

[2.2 New file saiqosmaps.h 1](#_Toc417378496)

[2.3 Changes to saiport.h 4](#_Toc417378497)

[2.3.1 New attributes 4](#_Toc417378498)

[2.4 Changes to saitypes.h 5](#_Toc417378499)

[2.5 Deprecate 5](#_Toc417378500)

[2.5.1 Changes to saiqos.h 5](#_Toc417378501)

[3 API Flow 6](#_Toc417378502)

[3.1 Trust Dot1p configuration example: 6](#_Toc417378503)

[3.1.1 Step 1: Create dot1p to tc map 6](#_Toc417378504)

[Step 2: Apply the created map on port to enable trust-dot1p: 7](#_Toc417378505)

[3.1.2 Create TC to Queue map 7](#_Toc417378506)

[Step 2: Apply the created map to port 9](#_Toc417378507)

[4 Appendix 9](#_Toc417378508)

[4.1 Review comments from Cavium: 9](#_Toc417378509)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for changes in QOS API |  | 02/19/15 |

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

QoS enables you to provide better service to certain flows. This document defines QOS object to create/remove/set and get the below maps.

Ingress Qos Maps:

Dot1P map: This profile holds the mapping of

Dot1p Priority -> Traffic Class

Dot1p Priority -> color

DSCP map: This profile holds the mapping of

DSCP -> Traffic Class

DSCP -> color

Egress Qos Maps:

Traffic class map: This profile holds the mapping of

Traffic class -> Egress queue

Traffic class -> DSCP

Traffic class -> DOT1P

Dot1p and DSCP maps are applied to in bound ports to classify the traffic based on trust dot1p/dscp configurations. Queue map is applied to out bound port to achieve the queuing.

# Specification

## Changes to sai.h

typedef enum \_sai\_api\_t

{

…

SAI\_API\_QOS\_MAP = 14,

} sai\_api\_t;

## New file saiqosmaps.h

/\*

\* Data Structire for below maps

\* Map DSCP to TC/color.

\* Map DOT1p to TC/color.

\* Map Tracffic class to Queue/DSCP/DOT1P.

\*/

typedef struct \_sai\_qos\_map\_t

{

Union {

sai\_cos\_t tc;

sai\_uint8\_t dscp;

sai\_uint8\_t dot1p;

} key;

union {

sai\_cos\_t tc;

sai\_packet\_color\_t color;

sai\_object\_id\_t qid;

sai\_uint8\_t dot1p;

sai\_uint8\_t dscp;

} value;

} sai\_qos\_map\_t;

typedef struct \_sai\_qos\_map\_list\_t

uint32\_t count;

sai\_qos\_map\_t \*list;

}sai\_qos\_map\_list\_t;

\* Qos Map Types

\* ASIC May support per port or global map tables and also

\* number of map tables suppoprted can be varied for each combination.

\* Ex: DOT1p-> Tc – 2 global maps

\* DOT1p-> Color – 3 global maps

\* DSCP-> Tc – 10 global maps

\* DOT1p-> Color – 3 global maps

\* SAI Design the maps seperetely.

\*/

typedef enum \_sai\_qos\_map\_type\_t {

/\* Qos Map to set DOT1P to Traffic class\*/

SAI\_QOS\_MAP\_DOT1P\_TO\_TC,

/\* Qos Map to set DOT1P to color\*/

SAI\_QOS\_MAP\_DOT1P\_TO\_COLOR,

/\* Qos Map to set DSCP to Traffic class\*/

SAI\_QOS\_MAP\_DSCP\_TO\_TC,

/\* Qos Map to set DSCP to color\*/

SAI\_QOS\_MAP\_DSCP\_TO\_COLOR,

/\* Qos Map to set traffic class to queue \*/

SAI\_QOS\_MAP\_TC\_TO\_QUEUE,

/\* Qos Map to set traffic class to DSCP \*/

SAI\_QOS\_MAP\_TC\_TO\_DSCP,

/\* Qos Map to set traffic class to DOT1P \*/

SAI\_QOS\_MAP\_TC\_TO\_DOT1P,

} sai\_qos\_map\_type\_t;

/\*

\* Attribute id for Qos Map

\*/

typedef enum \_sai\_qos\_map\_attr\_t

{

/\* READ-ONLY \*/

/\* READ-WRITE \*/

/\* Qos Map type [sai\_qos\_map\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_QOS\_MAP\_ATTR\_TYPE,

/\* Dot1p to TC Mapping [sai\_qos\_map\_list\_t],

\* Defaults:

\* All Dot1p/DSCP maps to traffic class 0

\* All Dot1p/DSCP maps to color SAI\_PACKET\_COLOR\_GREEN

\* All traffic class maps to queue 0.

\*/

SAI\_QOS\_MAP\_ATTR\_MAP\_TO\_VALUE\_LIST,

} sai\_qos\_map\_attr\_t ;

/\*

\* Routine Description:

\* Create Qos Map

\*

\* Arguments:

\* [out] qos\_map\_id - Qos Map 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\_qos\_map\_fn)(

\_Out\_ sai\_object\_id\_t\* qos\_map\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Remove Qos Map

\*

\* Arguments:

\* [in] qos\_map\_id – Qos Map id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_qos\_map\_fn) (

\_In\_ sai\_object\_id\_t qos\_map\_id,

);

/\*

\* Routine Description:

\* Set Qos Map attribute

\*

\* Arguments:

\* [in] qos\_map\_id – Qos Map Id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_qos\_map\_attribute\_fn)(

\_In\_ sai\_object\_id\_t qos\_map\_id,

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

);

\* Routine Description:

\* Get Qos Map attribute

\*

\* Arguments:

\* [in] qos\_map\_id – Qos Map Id

\* [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\_qos\_map\_attribute\_fn)(

\_In\_ sai\_object\_id\_t qos\_map\_id ,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Qos Map methods table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_qos\_map\_api\_t

{

sai\_create\_qos\_map\_fn create\_qos\_map;

sai\_remove\_qos\_map\_fn remove\_qos\_map;

sai\_set\_qos\_map\_attribute\_fn set\_qos\_map\_attribute;

sai\_get\_qos\_map\_attribute\_fn get\_qos\_map\_attribute;

} sai\_qos\_map\_api\_t;

## Changes to saiport.h

### New attributes

typedef enum \_sai\_port\_attr\_t

{

..

..

/\* READ-ONLY \*/

/\* READ-WRITE \*/

/\* Port default Traffic class Mapping \*/

SAI\_PORT\_ATTR\_QOS\_DEFAULT\_TC,

/\* Enable DOT1P -> TC MAP [sai\_object\_id\_t] on port

\* MAP id = SAI\_NULL\_OBJECT\_ID to disable map on port.

\* To enable/disbale trust Dot1p, Map ID should be add/remove on port \*/

SAI\_PORT\_ATTR\_QOS\_DOT1P\_TO\_TC\_MAP,

/\* Enable DOT1P -> COLOR MAP [sai\_object\_id\_t] on port

\* MAP id = SAI\_NULL\_OBJECT\_ID to disable map on port.

\* To enable/disbale trust Dot1p, Map ID should be add/remove on port \*/

SAI\_PORT\_ATTR\_QOS\_DOT1P\_TO\_COLOR\_MAP,

/\* Enable DSCP -> TC MAP [sai\_object\_id\_t] on port

\* MAP id = SAI\_NULL\_OBJECT\_ID to disable map on port.

\* To enable/disbale trust DSCP, Map ID should be add/remove on port \*/

SAI\_PORT\_ATTR\_QOS\_DSCP\_TO\_TC\_MAP,

/\* Enable DSCP -> COLOR MAP [sai\_object\_id\_t] on port

\* MAP id = SAI\_NULL\_OBJECT\_ID to disable map on port.

\* To enable/disbale trust DSCP, Map ID should be add/remove on port \*/

SAI\_PORT\_ATTR\_QOS\_DSCP\_TO\_COLOR\_MAP,

/\* Enable TC -> Queue MAP [sai\_object\_id\_t] on port

Map id = SAI\_NULL\_OBJECT\_ID to disable map on port \*/

SAI\_PORT\_ATTR\_QOS\_TC\_TO\_QUEUE\_MAP,

/\* Enable TC -> DOT1P MAP [sai\_object\_id\_t]

Map id = SAI\_NULL\_OBJECT\_ID to disable map on port \*/

SAI\_PORT\_ATTR\_QOS\_TC\_TO\_DOT1P\_MAP,

/\* Enable TC -> DSCP MAP [sai\_object\_id\_t]

Map id = SAI\_NULL\_OBJECT\_ID to disable map on port \*/

SAI\_PORT\_ATTR\_QOS\_TC\_TO\_DSCP\_MAP,

} sai\_port\_attr\_t;

## Changes to saitypes.h

typedef union {

sai\_qos\_map\_list\_t qosmap;

} sai\_attribute\_value\_t;

## Deprecate

### Changes to saiqos.h

typedef enum \_sai\_cos\_port\_trust\_t {

/\* Do not trust packet fields for setting CoS \*/

SAI\_COS\_PORT\_TRUST\_NONE,

/\* Trust packets 802.1p field for setting CoS \*/

SAI\_COS\_PORT\_TRUST\_DOT1P,

/\* Trust packets DSCP field for setting CoS \*/

SAI\_COS\_PORT\_TRUST\_DSCP

} sai\_cos\_port\_trust\_t;

/\*

\* Cos map type selector

\*/

typedef enum \_sai\_cos\_map\_t {

/\* Dot1P to Traffic class Mapping \*/

SAI\_COS\_MAP\_DOT1P\_TO\_TC,

/\* DSCP to Traffic Mapping \*/

SAI\_COS\_MAP\_DSCP\_TO\_TC,

/\* TC Mapping to schedule node mapping.

}

/\*

\* Routine Description:

\* Set "class of service" mapping for the port. In order for the mapping

\* to be active, the SAI\_COS\_ATTR\_PORT\_TRUST must be set.

\*

\* Arguments:

\* [in] port\_id - port id

\* [in] qos\_map\_selector - qos mapping type

\* [in] value\_to\_map - value, depends on qos mapping type (not used/801p/dscp)

\* [in] cos\_value - the mapped cos value.

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_cos\_mapping\_fn)(

\_In\_ sai\_port\_id\_t port\_id,

\_In\_ sai\_qos\_map\_t qos\_map\_selector,

\_In\_ uint32\_t value\_to\_map,

\_In\_ sai\_cos\_t cos\_value

);

/\*

\* Routine Description:

\* Get "class of service" mapping for the port. In order for the mapping

\* to be active, the SAI\_COS\_ATTR\_PORT\_TRUST must be set.

\*

\* Arguments:

\* [in] port\_id - port id

\* [in] qos\_map\_selector - qos mapping type

\* [in] value\_to\_map - value, depends on qos mapping type (not used/801p/dscp)

\* [in] cos\_value - the mapped cos value.

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_cos\_mapping\_fn)(

\_In\_ sai\_port\_id\_t port\_id,

\_In\_ sai\_qos\_map\_t qos\_map\_selector,

\_In\_ uint32\_t value\_to\_map,

\_Out\_ sai\_cos\_t\* cos\_value

);

# API Flow

## Trust Dot1p configuration example:

### Step 1: Create dot1p to tc map

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dot1p | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| TC | 1 | 0 | 2 | 3 | 4 | 5 | 6 | 7 |

sai\_object\_id\_t dot1p\_to\_tc\_map\_id;

sai\_attribute\_t sai\_attr;

int attr\_count = 2;

sai\_qos\_map\_list\_t dot1p\_to\_tc\_map\_list;

dot1p\_to\_tc\_map\_list.count = 8;

dot1p\_to\_tc\_map\_list.list =

(sai\_qos\_map\_t \*)calloc(dot1p\_to\_tc\_map\_list.count, sizeof(sai\_qos\_map\_t));

dot1p\_to\_tc\_map\_list.list[0].key.dot1p = 0;

dot1p\_to\_tc\_map\_list.list[0].value.tc = 1;

dot1p\_to\_tc\_map\_list.list[1].key.dot1p = 1;

dot1p\_to\_tc\_map\_list.list[1].value.tc = 0;

dot1p\_to\_tc\_map\_list.list[2].key.dot1p = 2;

dot1p\_to\_tc\_map\_list.list[2].value.tc = 2;

dot1p\_to\_tc\_map\_list.list[3].key.dot1p = 3;

dot1p\_to\_tc\_map\_list.list[3].value.tc = 3;

dot1p\_to\_tc\_map\_list.list[4].key.dot1p = 4;

dot1p\_to\_tc\_map\_list.list[4].value.tc = 4;

dot1p\_to\_tc\_map\_list.list[5].key.dot1p = 5;

dot1p\_to\_tc\_map\_list.list[5].value.tc = 5;

dot1p\_to\_tc\_map\_list.list[6].key.dot1p = 6;

dot1p\_to\_tc\_map\_list.list[6].value.tc = 6;

dot1p\_to\_tc\_map\_list.list[7].key.dot1p = 7;

dot1p\_to\_tc\_map\_list.list[7].value.tc = 7;

attr\_list[0].id = SAI\_QOS\_MAP\_ATTR\_TYPE;

attr\_list[0].value.s32 = SAI\_QOS\_MAP\_DOT1P\_TO\_TC;

attr\_list[1].id = SAI\_QOS\_MAP\_ATTR\_MAP\_TO\_VALUE\_LIST;

attr\_list[1].value.qosmap.count = dot1p\_to\_tc\_map\_list.count;

attr\_list[1].value.qosmap.list = dot1p\_to\_tc\_map\_list.list;

sai\_create\_qos\_map\_fn (&dot1p\_to\_tc\_map\_id, attr\_count, &attr\_list);

### Step 2: Apply the created map on port to enable trust-dot1p:

sai\_object\_id\_t port\_id;

sai\_attribute\_t sai\_port\_attr\_set;

sai\_port\_attr\_set.id = SAI\_PORT\_ATTR\_QOS\_DOT1P\_TO\_TC\_MAP;

sai\_port\_attr\_set.value.oid = dot1p\_to\_tc\_map\_id;

sai\_set\_port\_attribute\_fn (port\_id, &sai\_port\_attr\_set);

### Create TC to Queue map

Example for system havin the 8 unicast queues and 4 multicast queues.

#### Step 1: Get the SAI queue object id’s

sai\_object\_id\_t appl\_ucast\_qid\_to\_sai\_qId[8];

sai\_object\_id\_t appl\_mcast\_qid\_to\_sai\_qId[8];

/\* Get the SAI queue ID [sai\_object\_id\_t]. Look for detailed example in queue proposal. \*/

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TC | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Unicast Queue | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Multicast queue | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 |

#### Step 2: Create the TC-> Queue map

sai\_object\_id\_t tc\_to\_queue\_map\_id;

sai\_attribute\_t sai\_attr;

int attr\_count = 2;

sai\_qos\_map\_list\_t tc\_to\_queue\_map\_list;

tc\_to\_queue\_map\_list.count = 16;

tc\_to\_queue \_map\_list.list =

(sai\_qos\_map\_t \*)calloc(tc\_to\_queue\_map\_list.count, sizeof(sai\_qos\_map\_t));

tc\_to\_queue\_map\_list.list[0].key.tc = 0;

tc\_to\_queue\_map\_list.list[0].value.queue = appl\_ucast\_qid\_to\_sai\_qId[0];

tc\_to\_queue\_map\_list.list[1].key.tc = 0;

tc\_to\_queue\_map\_list.list[1].value.queue = appl\_mcast\_qid\_to\_sai\_qId[0];

tc\_to\_queue\_map\_list.list[2].key.tc = 1;

tc\_to\_queue\_map\_list.list[2].value.queue = appl\_ucast\_qid\_to\_sai\_qId[1];

tc\_to\_queue\_map\_list.list[3].key.tc = 1;

tc\_to\_queue\_map\_list.list[3].value.queue = appl\_mcast\_qid\_to\_sai\_qId[1];

tc\_to\_queue\_map\_list.list[4].key.tc = 2;

tc\_to\_queue\_map\_list.list[4].value.queue = appl\_ucast\_qid\_to\_sai\_qId[2];

tc\_to\_queue\_map\_list.list[5].key.tc = 2;

tc\_to\_queue\_map\_list.list[5].value.queue = appl\_mcast\_qid\_to\_sai\_qId[2];

tc\_to\_queue\_map\_list.list[6].key.tc = 3;

tc\_to\_queue\_map\_list.list[6].value.queue = appl\_ucast\_qid\_to\_sai\_qId[3];

tc\_to\_queue\_map\_list.list[7].key.tc = 3;

tc\_to\_queue\_map\_list.list[7].value.queue = appl\_mcast\_qid\_to\_sai\_qId[3];

tc\_to\_queue\_map\_list.list[8].key.tc = 4;

tc\_to\_queue\_map\_list.list[8].value.queue = appl\_ucast\_qid\_to\_sai\_qId[4];

tc\_to\_queue\_map\_list.list[9].key.tc = 4;

tc\_to\_queue\_map\_list.list[9].value.queue = appl\_mcast\_qid\_to\_sai\_qId[3];

tc\_to\_queue\_map\_list.list[10].key.tc = 5;

tc\_to\_queue\_map\_list.list[10].value.queue = appl\_ucast\_qid\_to\_sai\_qId[5];

tc\_to\_queue\_map\_list.list[11].key.tc = 5;

tc\_to\_queue\_map\_list.list[11].value.queue = appl\_mcast\_qid\_to\_sai\_qId[3];

tc\_to\_queue\_map\_list.list[12].key.tc = 6;

tc\_to\_queue\_map\_list.list[12].value.queue = appl\_mcast\_qid\_to\_sai\_qId[6];

tc\_to\_queue\_map\_list.list[13].key.tc = 6;

tc\_to\_queue\_map\_list.list[13].value.queue = appl\_mcast\_qid\_to\_sai\_qId[3];

tc\_to\_queue\_map\_list.list[14].key.tc = 7;

tc\_to\_queue\_map\_list.list[14].value.queue = appl\_mcast\_qid\_to\_sai\_qId[7];

tc\_to\_queue\_map\_list.list[15].key.tc = 7;

tc\_to\_queue\_map\_list.list[15].value.queue = appl\_mcast\_qid\_to\_sai\_qId[3];

attr\_list[0].id = SAI\_QOS\_MAP\_ATTR\_TYPE;

attr\_list[0].value.s32 = SAI\_QOS\_MAP\_TC\_TO\_QUEUE;

attr\_list[1].id = SAI\_QOS\_MAP\_ATTR\_MAP\_TO\_VALUE\_LIST;

attr\_list[1].value.qosmap.count = tc\_to\_queue\_map\_list.count;

attr\_list[1].value.qosmap.list = tc\_to\_queue\_map\_list.list;

sai\_create\_qos\_map\_fn (&tc\_to\_queue\_map\_id, attr\_count, &attr\_list);

### Step 2: Apply the created map to port

sai\_object\_id\_t port\_id;

sai\_attribute\_t sai\_port\_attr\_set;

sai\_port\_attr\_set.id = SAI\_PORT\_ATTR\_QOS\_TC\_TO\_QUEUE\_MAP;

sai\_port\_attr\_set.value.oid = tc\_to\_queue\_map\_id;

sai\_set\_port\_attribute\_fn (port\_id, &sai\_port\_attr\_set);

# Appendix

## Review comments from Cavium:

Hi Ashok,

Thanks for comments. I have no further comments at this point.

Regards,

Atit

From: Jain, Atit [<mailto:Atit.Jain@caviumnetworks.com>]   
Sent: Tuesday, March 17, 2015 12:53 PM  
To: Daparthi, Ashok; [gulv@microsoft.com](mailto:gulv@microsoft.com); [kasubra@microsoft.com](mailto:kasubra@microsoft.com)  
Cc: Manickam, Arunsubash; Natchimuth, Anbalagan; Shah, Zubin  
Subject: RE: SAI Subgroups Traffic policer in QoS

Hi Ashok,

On the SAI-Proposal-QOS-MAPS-Ver1. The understanding I have is this proposal defines:

Which map and how many entries supported per map

CoS sources per port

Mappings @ ingress and egress

Applying maps to ports

Following are the inputs that I have:

Optional: Some ASICs and systems allow mapping based on 802.1ad ie 802.1p and CFI/DEI

<Ashok D> For base we can do with the 802.1p alone. We can extended this by adding the map type as SAI\_QOS\_MAP\_DOT1P\_AND\_CFI/DEI\_TO\_TC in future.

Optional: Some ASIC and systems allow  both the sourced dot1p and DSCP with DSCP having more preference and dot1p only for non-l3 packets

<Ashok D> Yes with this proposal also we can have both the maps in same port, we can create DOT1p map and get X as id and DSCP map Y as id and by using the SAI\_PORT\_ATTR\_QOS\_MAP\_LIST attribute in port we can apply both.  Based on NPU behavior will take place as you mentioned. Added SAI\_PORT\_QOS\_TRUST\_BOTH

With below, I see 2 issues:

It doesn’t align to the common set signature

It should be added as part of port set API as we have already defined SOURCE, DEFAULT and MAP attributes there, so it would make a sense to add it there

typedef sai\_status\_t (\*sai\_set\_cos\_mapping\_fn)(

    \_In\_ sai\_port\_id\_t port\_id,

    \_In\_ sai\_qos\_map\_t qos\_map\_selector,

    \_In\_ uint32\_t value\_to\_map,

    \_In\_ sai\_cos\_t cos\_value

    );

<Ashok D>

        I want make it clear on #3.  section 2.6 is in remove list. We are not going to use above function. This functionality is achieved by configuring map and apply map id as port attribute.

SAI\_PORT\_ATTR\_QOS\_MAP\_LIST attribute is defined for ingress or egress, I think we need 2 maps there, one for ingress (Dxxx-TC/COLOR etc) and one for egress (TC->Dxxx)

<Ashok D> SAI\_PORT\_ATTR\_QOS\_MAP\_LIST attribute is to apply the list of maps. This is defined as below. So we can apply the Dxxx-TC, Dxxx-COLOR and TC->DSCP and TC->DOT1P.

typedef struct \_sai\_qos\_map\_list\_t {

    uint32\_t qos\_map\_count;

    sai\_qos\_map\_id\_t \*qos\_map\_list;

} sai\_qos\_map\_list\_t;

I still feel that in the egress we should use both TC/Color as match and map to Dxxx from there.

<Ashok D> For base we can do with the TC alone I.e TC + Any Color . We can extended this by adding the map type as SAI\_QOS\_MAP\_TC\_AND\_COLOR\_TO\_DSCP/DOT1P in future.

some minor ones in doc

Regards,

Atit

From: Jain, Atit [<mailto:Atit.Jain@caviumnetworks.com>]   
Sent: Thursday, March 12, 2015 12:45 PM  
To: Daparthi, Ashok; [gulv@microsoft.com](mailto:gulv@microsoft.com); [kasubra@microsoft.com](mailto:kasubra@microsoft.com)  
Cc: Manickam, Arunsubash; Natchimuth, Anbalagan  
Subject: RE: SAI Subgroups Traffic policer in QoS

HI Ashok,

Did a quick review, following are the comments, check doc for context:

Based on my understanding the marking of outgoing DSCP/Dot1p should happen at the egress. The sequence should be ingress(DSCP/dot1p-> TC/Color) egress (TC/Color -> DSCP/Dot1p). As I understand the idea to modify the DSCP/Dot1p is to let the next-hop/network know how to treat a packet, so this should be happening at the egress based on egress queue/port/flow etc.

<Ashok D> There are 2 options,

1# New Dot1p value can be selected from ingress and marked while packet goes out of egress port.

2#  Egress port can rewrite traffic class -> Dot1p/DSCP.

Both the cases remark will happen egress port,  but value can get from ingress map/egress maps in pipeline. Both maps are applied then egress will take more precedence.

I can add  new options

  traffic class -> DSCP

 Traffic class -> Dot1p

 Is cavium supports traffic class to Dot1p/DSCP?

BCM does not support this directly by any tables/register

<Atit> Sorry the actual order should be Dot1p/DSCP-> TC -> internal Queue ->Dot1p/DSCP. Broadcom XGS devices support it through EGRESS\_PRI\_CNG\_MAP table and I think for DSCP using some EGR\_DSCP\_TABLE. The index to these tables are indexed by port, internal queue and CNG. So I am sure of per port internal prio/CNG to dot1p and DSCP mapping in BCM. We had this per port for one FPGA based implementation. Supported the same on EzChip also.

We need to define egress cos mapping also where TC/Color is mapped to dot1p and DSCP

<Ashok D> We can add all those options, For base proposal I am planning have sub set of . I will remove the other options for v0.9.2. Do you feel below are enough.

                        DSCP -> Traffic class and color

                        Dot1p-> Traffic class and color

                        Traffic class -> queue.

I did not find TC-Queue mapping this in the doc? Sorry If I am missing something.

     <Ashok D> This is defined as (SAI\_QOS\_MAP\_ATTR\_TC\_TO\_QUEUE)

Can we merge this to saiqos.h to avoid too many files. (just a suggestion anyway is ok)

sai\_qos\_map\_attr\_t  can be made concise by keeping – ATTR\_SOURCE and ATTR\_MAP\_TO\_[TC, COLOR, DSCP etc] only and need not define twice for each trust type

<Ashok D>  This proposal has 3  maps for now.

 DSCP-> traffic class and color.

 DOT1P-> traffic class and color.

Traffic Class -> queue

We defined it as separate map by assuming some ASIC may support limited number of maps as  global tables and these maps can be applied it individual port after creation. So making it is as separate maps more sense instead of direct port settings.

In this case, I am thinking of ASIC may support tables separately for all below options

DSCP-> Traffic class

DSCP-> Color

DOT1P-> Traffic class

DOT1P-> Color

Traffic class -> Queue.

<Atit> I think following is good enough, can always be mapped to different tables if its required.

DSCP-> traffic class and color.

 DOT1P-> traffic class and color.

Traffic Class -> queue

So we need to create the maps  also separately.  I am thinking of modifying it as below.

typedef enum \_sai\_qos\_map\_type\_t  {

    /\* Qos Map to set DOT1P to CoS vlaues \*/

    SAI\_QOS\_MAP\_DOT1P\_TC,

    SAI\_QOS\_MAP\_DOT1P\_COLOR,

    /\* Qos Map to set DSCP to CoS vlaues \*/

    SAI\_QOS\_MAP\_DSCP\_TC,

    SAI\_QOS\_MAP\_DSCP\_COLOR,

    /\* Qos Map to set traffic class to queue \*/

    SAI\_QOS\_MAP\_TC\_QUEUE,

} sai\_qos\_map\_type\_t;

    /\* Qos Map type [sai\_qos\_map\_source\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

    SAI\_QOS\_MAP\_ATTR\_TYPE,

    /\* Map to value Mapping [sai\_qos\_dot1p/dcsp/tc\_to\_value\_list\_t],

        Default All Dot1p to 0 traffic class and green

        Default All DSCP to 0 traffic class and green

        Default All traffic class to queue 0  \*/

     SAI\_QOS\_MAP\_ATTR\_MAP\_TO\_VALUE,

Please let me know your vies on this.

Is \_sai\_cos\_port\_trust\_t different from \_sai\_port\_qos\_trust\_t  could we keep only one

Is \_sai\_cos\_port\_trust\_t  different from \_sai\_port\_qos\_trust\_t  could we keep only one

sai\_set\_cos\_mapping\_fn signature is different from other other apis.

<Ashok D> 6,7,8 are deprecated.

If required we can have a quick call. Also, please let me know if I am added to subgroup, my organization needs to be indicated about it.

Regards,  
Atit