

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **L2MC** |
| **Authors** | **CENTEC** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **17/10/2015** |
| **SAI-Version** | **0.9.3** |

**Contents**

[List of Changes i](#_Toc459908587)

[1 Overview 1](#_Toc459908588)

[2 Specification 1](#_Toc459908589)

[2.1 Changes to saiswitch.h 1](#_Toc459908590)

[2.2 Changes to sai.h 1](#_Toc459908591)

[2.3 Changes to saivlan.h 1](#_Toc459908592)

[2.4 New definitions in sail2mc.h 2](#_Toc459908593)

[3 Examples 4](#_Toc459908594)

[3.1 Create a generic group 4](#_Toc459908595)

[3.2 Create group members 4](#_Toc459908596)

[3.3 Create L2MC entry 5](#_Toc459908597)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| Initial Version | Base version |  | 17/10/2015 |
| 0.2 | Changes after meeting review |  | 01/07/2016 |
| 0.6 | Extend FDB to implement L2Multicast | Min Yao | 20/07/2016 |
| 0.7 | Add NEXTHOP GROUPID in vlan entry attribute as per internal discussion | Min Yao | 04/08/2016 |
| 0.8 | * Add Switch-level attribute to define whether switch supports IP based L2 multicast. (RO) | Min Yao | 25/08/2016 |

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 chL2 I/O and driver technology, bus technology, memory chL2 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

L2 Multicast, a basic layer 2 feature, which provides a security solution by isolating flooding domain from VLAN to l2multicast group. Meanwhile, it decreases the traffic load on a Layer 2 switch. L2 multicast can be implemented by multicast fdb lookup table (referred to as MAC based L2 multicast) or SG/\*G lookup table (referred to as IP based L2 multicast, known as snooping). This document covers the implementation of IP based L2 multicast.

# Specification

This section describes the details of the proposed interface/API

## Changes to saiswitch.h

/\*\*

\* Attribute Id in sai\_set\_switch\_attribute() and

\* sai\_get\_switch\_attribute() calls

\*/

typedef enum \_sai\_switch\_attr\_t

{

SAI\_SWITCH\_ATTR\_START,

/\*\* READ-ONLY \*/

…

**/\*\* Multicast snooping capability supported by the NPU. [bool]**

**\* Value true indicates that the NPU is capable of performing SG/\*G lookup for L2 multicast (IP based L2 multicast)**

**\* Value false indicates that the NPU is only capable of performing multicast fdb lookup for L2 multicast (MAC based L2 mulitcast)\*/**

**SAI\_SWITCH\_ATTR\_MULTICAST\_SNOOPING\_CAPABILITY,**

…

} sai\_switch\_attr\_t;

## Changes to sai.h

typedef enum \_sai\_api\_t

{

…

**SAI\_API\_L2MC= 44, /\* sai\_l2mc\_api\_t \*/**

**…**

} sai\_api\_t;

## Changes to saivlan.h

/\*\*

\* @brief Attribute Id in sai\_set\_vlan\_attribute() and

\* sai\_get\_vlan\_attribute() calls

\*/

typedef enum \_sai\_vlan\_attr\_t

{

/\*\* READ-WRITE \*/

**…**

**/\*\* VLAN IGMP Snooping enable [bool]**

**\*(default is enabled)\*/**

**SAI\_VLAN\_ATTR\_IGMP\_SNOOPING\_EN,**

**/\*\* VLAN MLD Snooping enable [bool]**

**\*(default is enabled)\*/**

**SAI\_VLAN\_ATTR\_MLD\_SNOOPING\_EN,**

…

/\*\* Custom range base value \*/

SAI\_VLAN\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_vlan\_attr\_t;

## New definitions in sail2mc.h

/\*\*

\* @brief Enum defining l2mc entry types.

\*/

typedef enum \_sai\_l2mc\_entry\_type\_t

{

/\*\* L2MC entry with type (S,G) \*/

SAI\_L2MC\_TYPE\_SG = 0x00000001,

/\*\* L2MC entry with type (\*,G) \*/

SAI\_L2MC\_TYPE\_XG = 0x00000002,

} sai\_l2mc\_entry\_type\_t;

/\*

\* L2 multicast entry key

\*/

typedef struct \_sai\_l2mc\_entry\_t

{

sai\_vlan\_id\_t vlan\_id;

sai\_l2mc\_entry\_type\_t type;

sai\_ip\_address\_t destination;

sai\_ip\_address\_t source;

} sai\_l2mc\_entry\_t;

/\*

\* Attribute Id for L2 multicast entry

\*/

typedef enum \_sai\_l2mc\_entry\_attr\_t

{

SAI\_L2MC\_ENTRY\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* READ-WRITE \*/

/\*\* Packet action [sai\_packet\_action\_t]

(default to SAI\_PACKET\_ACTION\_FORWARD) \*/

SAI\_L2MC\_ATTR\_PACKET\_ACTION = SAI\_L2MC\_ENTRY\_ATTR\_START,,

/\*\* Output group id for the packet [sai\_object\_id\_t]

\* The group type should be SAI\_GROUP\_TYPE\_L2MC

\* (MANDATORY\_ON\_CREATE when SAI\_L2MC\_ATTR\_PACKET\_ACTION == SAI\_PACKET\_ACTION\_FORWARD)

\* (CREATE\_AND\_SET) \*/

SAI\_L2MC\_ATTR\_OUTPUT\_GROUP\_ID,

/\* Custom range base value \*/

SAI\_L2MC\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_l2mc\_entry\_attr\_t;

/\*

\* Routine DescrL2tion:

\* Create L2 multicast entry

\*

\* Arguments:

\* [in] l2mc\_entry - L2 multicast entry

\* [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\_l2mc\_entry\_fn)(

\_In\_ const sai\_l2mc\_entry\_t\* l2mc\_entry,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine DescrL2tion:

\* Remove L2 multicast entry

\*

\* Arguments:

\* [in] l2mc\_entry - L2 multicast entry

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_l2mc\_entry\_fn)(

\_In\_ const sai\_l2mc\_entry\_t\* l2mc\_entry

);

/\*

\* Routine DescrL2tion:

\* Set L2 multicast entry attribute value

\*

\* Arguments:

\* [in] L2 multicast - L2 multicast entry

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_l2mc\_entry\_attribute\_fn)(

\_In\_ const sai\_l2mc\_entry\_t\* l2mc\_entry,

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

);

/\*

\* Routine DescrL2tion:

\* Get L2 multicast entry attribute value

\*

\* Arguments:

\* [in] l2mc\_entry - L2 multicast entry

\* [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\_l2mc\_entry\_attribute\_fn)(

\_In\_ const sai\_l2mc\_entry\_t\* l2mc\_entry,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* L2 multicast method table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_l2mc\_api\_t

{

sai\_create\_l2mc\_entry\_fn create\_l2mc\_entry;

sai\_remove\_l2mc\_entry\_fn remove\_l2mc\_entry;

sai\_set\_l2mc\_entry\_attribute\_fn set\_l2mc\_entry\_attribute;

sai\_get\_l2mc\_entry\_attribute\_fn get\_l2mc\_entry\_attribute;

} sai\_l2mc\_api\_t;

# Examples

Create a generic group with multiple output port and associate the group to a multicast fdb entry.

## Create a generic group

sai\_api\_query(SAI\_API\_GROUP, &group\_api);

sai\_object\_id\_t l2mc\_group\_id = 0;

sai\_attribute\_t attr[1] = {0};

attr[0].id = SAI\_GROUP\_ATTR\_TYPE;

attr[0].value.s32 = SAI\_GROUP\_TYPE\_L2MC;

group\_api->create\_group(&l2mc\_group\_id, 1, attr);

## Create group members

sai\_object\_id\_t l2mc\_group\_member\_id1 = 0;

sai\_object\_id\_t l2mc\_group\_member\_id2 = 0;

sai\_attribute\_t attr[2] = {0};

attr[0].id = SAI\_GROUP\_MEMBER\_ATTR\_GROUP\_ID;

attr[0].value.oid = l2mc\_group\_id;

attr[1].id = SAI\_GROUP\_MEMBER\_ATTR\_MEMBER\_ID;

attr[1].value.oid = port1\_obj; /\* some valid port object\*/

group\_api->create\_group\_member (&l2mc\_group\_member\_id1, 2, attr);

attr[0].id = SAI\_GROUP\_MEMBER\_ATTR\_GROUP\_ID;

attr[0].value.oid = l2mc\_group\_id;

attr[1].id = SAI\_GROUP\_MEMBER\_ATTR\_MEMBER\_ID;

attr[1].value.oid = port2\_obj; /\* some valid port object\*/

group\_api->create\_group\_member(&l2mc\_group\_member\_id2, 2, attr);

## Create L2MC entry

sai\_api\_query(SAI\_API\_L2MC, &sai\_l2mc\_api);

sai\_l2mc\_entry\_t l2mc\_entry;

sai\_attribute\_t attr[2] = {0};

l2mc\_entry.vlan\_id = vlan\_id\_obj;

l2mc\_entry.type = SAI\_L2MC\_TYPE\_XG

l2mc\_entry.destination.addr\_family = SAI\_L2\_ADDR\_FAMILY\_IPV4;

l2mc\_entry.destination.addr.ip4 = 0xE0010101;

attr[0].id = SAI\_L2MC\_ATTR\_PACKET\_ACTION;

attr[0].value.s32 = SAI\_PACKET\_ACTION\_FORWARD;

attr[1].id = SAI\_L2MC\_ATTR\_OUTPUT\_GROUP\_ID;

attr[1].value.oid = l2mc\_group\_id;

sai\_l2mc\_api->create\_l2mc\_entry(&L2mc\_entry, 2, attr);