

|  |
| --- |
|  |

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](#_Toc456995139)

[1 Overview 1](#_Toc456995140)

[2 Specification 1](#_Toc456995141)

[2.1 Change to sainexthopgroup.h 1](#_Toc456995142)

[2.2 Changes to saifdb.h 1](#_Toc456995143)

[2.3 Changes to saivlan.h 2](#_Toc456995144)

[3 Examples 3](#_Toc456995145)

[4 Appendix (review meeting questions) 4](#_Toc456995146)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.3 | Base version |  | 17/10/2015 |
| 0.9.3 | Changes after meeting review |  | 01/07/2016 |
| 0.9.3 | Extend FDB to implement L2Multicast | Min Yao | 20/07/2016 |

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 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

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. This document covers details of l2multicast object –l2multicast group creating and removing, l2multicast group attributes setting and getting, port adding to l2multicast group, and port removing from l2multicast group.

# Specification

This section describes the details of the proposed interface/API

## Change to sainexthopgroup.h

/\*

\* Next hop group type

\*/

typedef enum \_sai\_next\_hop\_group\_type\_t

{

/\*\* Next hop group is ECMP \*/

SAI\_NEXT\_HOP\_GROUP\_ECMP,

**/\*\* Next hop group is Layer2 multicast, group member could be a port/LAG/L2tunnel \*/**

**SAI\_NEXT\_HOP\_GROUP\_L2MC,**

/\* Other types of next hop group to be defined in the future, e.g., WCMP \*/

} sai\_next\_hop\_group\_type\_t;

Add a nexthop group type SAI\_NEXT\_HOP\_GROUP\_L2MC . Use l2mc nexthop group to manage the out if list. L2mc nexthop group member could be port, LAG and L2 tunnel.

## Changes to saifdb.h

/\*\*

\* @brief Attribute Id for fdb entry

\*/

typedef enum \_sai\_fdb\_entry\_attr\_t

{

SAI\_FDB\_ENTRY\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* READ-WRITE \*/

/\*\* FDB entry type [sai\_fdb\_entry\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET) \*/

SAI\_FDB\_ENTRY\_ATTR\_TYPE = SAI\_FDB\_ENTRY\_ATTR\_START,

~~/\*\* FDB entry port id [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET)~~

~~\* The port id here can refer to a generic port object such as SAI port object id,~~

~~\* SAI LAG object id and etc. or to a tunnel next hop object in case the entry is~~

~~\* l2 tunnel \*/~~

~~SAI\_FDB\_ENTRY\_ATTR\_PORT\_ID,~~

/\*\* FDB entry packet action [sai\_packet\_action\_t] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET) \*/

SAI\_FDB\_ENTRY\_ATTR\_PACKET\_ACTION,

**/\*\* FDB entry desination id [sai\_object\_id\_t]**

**\* The destination id here can refer to a generic port object such as SAI port object id,**

**\* SAI LAG object id and etc. or to a tunnel next hop object in case the entry is**

**\* l2 tunnel**

**\* It can also refer to a nexthop group id with type SAI\_NEXT\_HOP\_GROUP\_L2MC**

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

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

**SAI\_FDB\_ENTRY\_ATTR\_DESTINATION\_ID,**

/\*\* User based Meta Data

\* [sai\_uint32\_t] (CREATE\_AND\_SET)

\* Value Range SAI\_SWITCH\_ATTR\_FDB\_DST\_USER\_META\_DATA\_RANGE \*/

SAI\_FDB\_ENTRY\_ATTR\_META\_DATA,

/\* -- \*/

SAI\_FDB\_ENTRY\_ATTR\_END,

/\* Custom range base value \*/

SAI\_FDB\_ENTRY\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\* --\*/

SAI\_FDB\_ENTRY\_ATTR\_CUSTOM\_RANGE\_END

} sai\_fdb\_entry\_attr\_t;

Associate a l2mc nexthop group to a fdb entry to perform packet replication.

## Changes to saivlan.h

/\*\*

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

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

\*/

typedef enum \_sai\_vlan\_attr\_t

{

SAI\_VLAN\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* List of vlan members in a VLAN [sai\_object\_list\_t]\*/

SAI\_VLAN\_ATTR\_MEMBER\_LIST = SAI\_VLAN\_ATTR\_START,

/\*\* READ-WRITE \*/

/\*\* Maximum number of learned MAC addresses [uint32\_t]

\* zero means learning limit disable. (default to zero).

\*/

SAI\_VLAN\_ATTR\_MAX\_LEARNED\_ADDRESSES,

/\*\* STP Instance that the VLAN is associated to [sai\_object\_id\_t]

\* (default to default stp instance id)\*/

SAI\_VLAN\_ATTR\_STP\_INSTANCE,

/\*\* To disable learning on a VLAN. [bool] (CREATE\_AND\_SET)

\* (default set to false)

\* This should override port learn settings. If this is set to true on a vlan,

\* then the source mac learning is disabled for this vlan on a member port even

\* if learn is enable on the port(based on port learn attribute)

\*/

SAI\_VLAN\_ATTR\_LEARN\_DISABLE,

**/\*\* VLAN Exception IGMP packet enable [bool]**

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

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

**/\*\* VLAN Exception MLD packet enable [bool]**

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

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

/\*\* User based Meta Data

\* [sai\_uint32\_t] (CREATE\_AND\_SET)

\* Value Range SAI\_SWITCH\_ATTR\_VLAN\_USER\_META\_DATA\_RANGE \*/

SAI\_VLAN\_ATTR\_META\_DATA,

SAI\_VLAN\_ATTR\_END,

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

SAI\_VLAN\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\* --\*/

SAI\_VLAN\_ATTR\_CUSTOM\_RANGE\_END

} sai\_vlan\_attr\_t;

# Examples

Create a l2mc nexthop group with multiple output port and associate the l2mc nexthop group with mcast fdb entry.

sai\_api\_query(SAI\_API\_NEXT\_HOP\_GROUP, &next\_hop\_group\_api);

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

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

sai\_object\_id\_t port\_list\_obj[4] = {0};

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

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

attr[1].id = SAI\_NEXT\_HOP\_GROUP\_ATTR\_NEXT\_HOP\_LIST;

port\_list\_obj [0] = port1\_obj; /\* some valid port object\*/

port\_list\_obj [1] = port2\_obj;

port\_list\_obj [2] = port3\_obj;

port\_list\_obj [3] = port4\_obj;

attr[1].value.objlist.count = 4;

attr[1].value.objlist.list = port\_list\_obj;

next\_hop\_group\_api->create\_next\_hop\_group (&l2mc\_nexthop\_group\_id, 2, attr);

sai\_api\_query(SAI\_API\_FDB, &sai\_fdb\_api);

sai\_fdb\_entry\_t fdb\_entry;

sai\_attribute\_t attr[3] = {0};

fdb\_entry.vlan\_id = 10;

fdb\_entry.mac\_address = {0x01, 0x00, 0x5e, 0x00, 0x02, 0x00}; /\* multicast\*/

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

attr[0].value.s32 = SAI\_FDB\_ENTRY\_STATIC;

attr[1].id = SAI\_FDB\_ENTRY\_ATTR\_PACKET\_ACTION;

attr[1].value.s32 = SAI\_PACKET\_ACTION\_FORWARD;

attr[2].id = SAI\_FDB\_ENTRY\_ATTR\_DESTINATION\_ID;

attr[2].value.obj = l2mc\_nexthop\_group\_id;

sai\_fdb\_api-> sai\_create\_fdb\_entry\_fn (&fdb\_entry, 3, attr);

# Appendix (review meeting questions)

1. Suggest the multicast group entry to be pointers to a group structure instead of port itself. That group structure could be shared by L2 and L3 multicast, containing the port bitmap for L2 and maybe VLAN info for L3.

Answer: for L3 multicast, please refer to <<SAI-Proposal-15-IP-Multicast.docx>>.

**Update[7/22] Now use nexthop group to manage l2 multicast members. We can also use nexthop group to manage l3 multicast members later.**

1. The port in the multicast group should consider tunnel port and bridge port both.

Answer: Yes, I will add it.

**Update[7/22] The L2mc nexthop group contains a list of sai objects, which could possibly be a port/LAG/l2 tunnel object.**

1. Regarding IGMP\_Disable flag (bool), how we defined the behavior when enabling it?

Answer: when IGMP\_SNOOPING\_EN is true and IGMP\_SNOOPING\_DISCARD\_UNKNOWN is true, the unknown traffic should be discard. When IGMP\_SNOOPING\_EN is true and IGMP\_SNOOPING\_DISCARD\_UNKNOWN is false, the unknown traffic should be flooding in VLAN.

**Update[7/22] Changed the flag to DISCARD\_UNKNOWN\_MCAST according to guohan’s suggestion.**

1. Is IGMP v3 considered in the proposal?

Answer: This proposal only consider L2 entry based on MAC and there are no consideration about L2 entry based on IP. For IGMP v3 snooping, according to the current usage, we suggest the software process v3 packets as v2. If there are request for L2 entry based on IP, we will add them in this proposal.

**Update[7/22] Need to discuss the behavior of IGMP snooping in bridging first. Will update later..**