

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **SAI Failover** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **24/12/2015** |
| **SAI-Version** | **V0.9.4** |

**Contents**

[List of Changes i](#_Toc438736857)

[1 Overview 1](#_Toc438736858)

[2 Proposal 1](#_Toc438736859)

[2.1 Failover Group and new attributes 1](#_Toc438736860)

[2.2 Associating Failover Group in FDB entry 2](#_Toc438736861)

[2.3 Associating Failover Group in Neighbor entry 2](#_Toc438736862)

[2.4 Associating Failover Group in Route entry 2](#_Toc438736863)

[3 Specification 2](#_Toc438736864)

[3.1 New file saifailover.h 2](#_Toc438736865)

[3.1.1 New attributes 2](#_Toc438736866)

[3.2 Changes to saifdb.h 4](#_Toc438736867)

[3.3 Changes to saineighbor.h 5](#_Toc438736868)

[3.4 Changes to sairoute.h 5](#_Toc438736869)

[4 Configuration Example 5](#_Toc438736870)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.4 | Proposal for SAI Failover |  | 24/12/2015 |

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 proposal defines a Failover Group object in SAI and enables switching the unicast traffic destined to a Primary SAI object to a predefined Backup SAI object on failure conditions. The mechanism for tracking the Primary object and triggering failover (health monitoring) is not proposed in SAI and it is in the scope of the Host Adapter in this proposal. For e.g, Host Adapter can use the SAI Failover Group for local link faults based on Link Monitoring or for IPv4 network failure based on BFD protocol session etc.

# Proposal



In the above multi-node redundant network topology, there is a backup path through the node B for the traffic between the hosts in Network 1 and Network 2.

When the link between nodes A and C goes down, traffic can be forwarded through the backup link. However, the MAC entries and ARP entries have to be flushed/aged and re-learnt through the backup link. In environments with many directly attached hosts, the number of entries can be high and this can cause a significant transition delay during which traffic might be dropped.

SAI Failover Group API can be used to leverage any capability in the ASIC for redirecting traffic towards the backup path in this scenario.

## Failover Group and new attributes

Host Adapter can define a Failover Group object containing a Primary SAI object and backup SAI object. This proposal covers the Port, LAG, RIF and NextHop objects in the Failover Group and it can be extended for other object types in future.

Failover Group defined with Port or LAG object can be associated to SAI FDB entry to protect the L2 unicast traffic destined to the FDB entry and forwarded to the port or LAG.

Failover Group defined with RIF object can be associated to SAI Neighbor entry to protect the L3 unicast traffic destined to the Neighbor.

Failover Group defined with NextHop object can be associated to SAI Route entry to protect the L3 unicast traffic destined to the route.

Application can set the failover status as active when the Primary object within a Failover Group is down or failed.

For multipath objects like LAG and NextHopGroup, there will be standard mechanism in the application to remove a member from the group, when the member fails. This will enable the traffic failover to another active member within the group – to another member port in the LAG or to another nexthop in the NextHopGroup. So, Failover within a LAG or NextHopGroup is not covered using the Failover Group defined in this proposal.

## Associating Failover Group in FDB entry

Failover Group object id can be set in the **SAI\_FDB\_ENTRY\_ATTR\_PORT\_ID** attribute to setup protection for the FDB entry. For e.g, Host Adapter can setup the Failover Group with backup object pointing to the Inter-Chassis Link for FDB entries learnt on a MLAG.

When Failover is set active, all FDB entries associated with the Failover Group will be forwarding traffic to the backup object.

NOTE1: Host Adapter has to ensure that the backup port is part of the VLAN to meet the VLAN filtering rules for switching the L2 traffic.

## Associating Failover Group in Neighbor entry

Failover Group object id can be passed in the **sai\_neighbor\_entry\_t** structure in the **rif\_id** parameter to setup protection for the IP Neighbor entry.

When Failover is set active, all Neighbor entries learnt on the Primary RIF will be forwarding traffic to the backup RIF object.

Further, SAI Next Hop can be created for the Neighbor using the Neighbor IP address and RIF Id as the Failover Group object id.

## Associating Failover Group in Route entry

Failover Group object id can be set in the **SAI\_ROUTE\_ATTR\_NEXT\_HOP\_ID** attribute to setup protection for the Route entry.

When Failover is set active, all Route entries associated with the Group will be forwarding traffic to the backup nexthop object.

# Specification

## New file saifailover.h

### New attributes

/\*\* Failover Group mode \*/

typedef enum \_sai\_failover\_mode\_t {

/\*\* Failure detection and Failover to be triggered from Host Adapter. \*/

SAI\_FAILOVER\_MODE\_SOFTWARE,

/\*\* Failure detection and Failover to be triggered in Hardware. \*/

SAI\_FAILOVER\_MODE\_HARDWARE,

} sai\_failover\_mode\_t;

typedef enum \_sai\_failover\_group\_attr\_t

{

/\*\* READ\_WRITE \*/

/\*\* Failover Mode. [sai\_failover\_mode\_t] (CREATE\_ONLY).

\* (default to SAI\_FAILOVER\_MODE\_SOFTWARE) \*/

SAI\_FAILOVER\_GROUP\_ATTR\_MODE,

/\*\* Primary object. [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY). \*/

SAI\_FAILOVER\_GROUP\_ATTR\_PRIMARY\_OBJECT,

/\*\* Backup object. [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY). \*/

SAI\_FAILOVER\_GROUP\_ATTR\_BACKUP\_OBJECT,

/\*\* Set Failover to enabled/disabled on the failover group. [bool].

Enabling failover will switch traffic to the backup object.

Disabling failover will revert traffic to the primary object. \*/

SAI\_FAILOVER\_GROUP\_ATTR\_FAILOVER\_ACTIVE,

} sai\_failover\_group\_attr\_t;

/\* Routine Description:

\* @ brief Create SAI Failover Group

\*

\* Arguments:

\* @param[out] failover\_group\_id – Failover Group id

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

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

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*

\*/

typedef sai\_status\_t (\*sai\_create\_failover\_group\_fn)(

\_Out\_ sai\_object\_id\_t\* failover\_group\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* Routine Description:

\* @brief Remove SAI Failover Group

\*

\* Arguments:

\* @param[in] failover\_group\_id – Failover Group id

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_failover\_group\_fn)(

\_In\_ sai\_object\_id\_t failover\_group\_id

);

/\* Routine Description:

\* @ brief Set Failover Group attribute

\*

\* Arguments:

\* @param[in] failover\_group\_id – Failover Group id

\* @param[in] attr - attribute

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_failover\_group\_attribute\_fn)(

\_In\_ sai\_object\_id\_t failover\_group\_id,

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

);

/\*\*

\* Routine Description:

\* @brief Get Failover Group attribute value

\*

\* Arguments:

\* @param[in] failover\_group\_id – Failover Group id

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

\* @param[inout] attrs - array of attributes

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_failover\_group\_attribute\_fn)(

\_In\_ sai\_object\_id\_t failover\_group\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\* SAI Failover Group method table retreived by sai\_api\_query

typedef struct \_sai\_failover\_group\_api\_t

{

sai\_create\_failover\_group\_fn create\_failover\_group;

sai\_remove\_failover\_group\_fn remove\_failover\_group;

sai\_set\_failover\_group\_attribute\_fn set\_failover\_group\_attribute;

sai\_get\_failover\_group\_attribute\_fn get\_failover\_group\_attribute;

} sai\_failover\_group\_api\_t;

## Changes to saifdb.h

/\*\* 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. on.

\* Associate to a Failover Group object containing the port and a backup object for setting up failover path \*/

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

## Changes to saineighbor.h

/\*\*

\* @brief neighbor entry

\*/

typedef struct \_sai\_neighbor\_entry\_t

{

/\* Associate to a Failover Group object containing the RIF and a backup RIF object for setting up failover path for the neighbor entry \*/

sai\_object\_id\_t rif\_id;

sai\_ip\_address\_t ip\_address;

} sai\_neighbor\_entry\_t;

## Changes to sairoute.h

/\*\* Next hop or next hop group id for the packet or a router interface

\* in case of directly reachable route [sai\_object\_id\_t]

\* Associate to a Failover Group object containing the Next hop and a backup Next hop object for setting up failover path \*/

SAI\_ROUTE\_ATTR\_NEXT\_HOP\_ID,

# Configuration Example

Following example shows how to setup failover for a L2 FDB entry learnt on a port with a backup LAG.

// Create a Failover Group for protecting port\_id\_1

sai\_attribute\_t attr[3];

attr[0].id = (sai\_attr\_id\_t) SAI\_FAILOVER\_GROUP\_ATTR\_MODE;

attr[0].value.booldata = false;

attr[1].id = (sai\_attr\_id\_t) SAI\_FAILOVER\_GROUP\_ATTR\_PRIMARY\_OBJECT;

attr[1].value.oid = port\_id\_1;

attr[2].id = (sai\_attr\_id\_t) SAI\_FAILOVER\_GROUP\_ATTR\_BACKUP\_OBJECT;

attr[2].value.oid = backup\_lag\_id;

sai\_failover\_group\_api->create\_failover\_group (&failover\_group1\_id, 3, attr);

// Attach Failover Group to the FDB entry learnt on port\_id\_1 to setup a protection path for the FDB entry.

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

sai\_mac\_t dummy\_mac\_addr = {0x0, 0x0, 0x0, 0x1, 0x2, 0x3};

unsigned int dummy\_vlan\_id = 100;

fdb\_attr[0].id = (sai\_attr\_id\_t) SAI\_FDB\_ENTRY\_ATTR\_TYPE

fdb\_attr[0].value.s32 = SAI\_FDB\_ENTRY\_DYNAMIC;

fdb\_attr[1].id = (sai\_attr\_id\_t) SAI\_FDB\_ENTRY\_ATTR\_PORT\_ID;

fdb\_attr[1].value.oid = failover\_group1\_id;

memcpy (fdb\_entry.mac\_address, dummy\_mac\_addr, sizeof (sai\_mac\_t));

fdb\_entry.vlan\_id = dummy\_vlan\_id;

sai\_fdb\_api->create\_fdb\_entry (&fdb\_entry, 2, fdb\_attr);

// Enable Failover

// Traffic is expected to be switched to backup LAG object after enabling failover.

sai\_attribute\_t attr[1];

attr[0].id = (sai\_attr\_id\_t) SAI\_FAILOVER\_GROUP\_ATTR\_FAILOVER\_ACTIVE;

attr[0].value.booldata = true;

sai\_failover\_group\_api->set\_failover\_group\_attribute (failover\_group1\_id, attr);