

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | Fast Reroute |
| **Authors** | Metaswitch Networks |
| **Status** | In Review |
| **Type** | Standards Track |
| **Created** | 13/04/2017 |
| **SAI-Version** | 1.1.0 |

Contents

[List of Changes 3](#_Toc481574671)

[1 Overview 5](#_Toc481574672)

[1.1 Overview of IP FRR 5](#_Toc481574673)

[1.2 Example 5](#_Toc481574674)

[1.3 Failure detection and switchover triggers 6](#_Toc481574675)

[1.3.1 Switchover in control plane vs switchover in switching entity 6](#_Toc481574676)

[2 Proposal 7](#_Toc481574677)

[2.1 Current SAI object model 7](#_Toc481574678)

[2.2 Proposed extensions 7](#_Toc481574679)

[2.2.1 Next Hop Group 7](#_Toc481574680)

[2.2.2 Next Hop Group Member 8](#_Toc481574681)

[3 Specification 9](#_Toc481574682)

[3.1 Addition to file sainexthopgroup.h 9](#_Toc481574683)

[3.1.1 Data Structures and Enumerations 9](#_Toc481574684)

[3.1.2 API 12](#_Toc481574685)

[4 Examples 12](#_Toc481574686)

[5 Pipeline 12](#_Toc481574687)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 1 | Base Version | Jonathan Hardwick, Michael Siuda | 12/04/2017 |
| 2 |  |  |  |

License

© 2017 Microsoft Corporation, Dell Inc., Facebook, Inc, Broadcom Limited, Intel Corporation, Marvell Semiconductor Ltd., Mellanox Technologies Ltd, Cavium, Centec, Barefoot, Metaswitch Networks.

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

Many IP/MPLS networks provide services which are very sensitive to traffic loss, which can occur when a node or link physically fails in the network (or software running on those nodes fails). For these networks, it is not acceptable to wait for protocol reconvergence to complete before reprogramming the Data Plane – applications may be sensitive to traffic loss greater than 10s of milliseconds. Technologies like IP FRR, which allow backup routes to be pre-programmed in the routers’ FIBs can significantly reduce the time it takes for a node to recover after a failure has been detected. This proposal therefore extends the SAI to allow pre-programming backup paths which is required for a device to support IP FRR.

## Overview of IP FRR

When there is a link or a node failure in an IP network, it can take several seconds for the Control Plane stack to converge the routers' FIBs to a new, consistent state which avoids the failed resource. During this time, the routers will still forward some packets along paths that include the failed resources, and as a result those packets will be black-holed. For some types of customer traffic (for example, real-time collaboration applications and pseudowires) this is too long an outage, and adversely affects the customer experience.

IP Fast Reroute (FRR) is intended to cut the window during which packets are black-holed down to tens of milliseconds. It works as follows.

* The routers compute a "safe" alternate next hop for each route in advance, to be used in case the primary next hop fails. See the example below for explanation what a “safe” alternative is.
* The alternate next hop is then pre-loaded into the Switching Entity so that the time it takes the router to switch over to the alternate can be minimized.

This mechanism allows packets to continue to reach their destinations while the Control Plane converges. Once the Control Plane has converged, the routers start to forward using the new primary next hops of all affected routes, and they compute a new set of alternate next hops, ready for the next failure.

## Example

Let’s consider the following network topology as an example. In this scenario router S programs its Switching Entity to forward traffic destined for node D.



As the first step the Control Plane stack on S calculates shortest path to D, which in this example is via node C. This is programmed to the Switching Entity.

Then, Control Plane calculates the backup next hop. In this topology there are two possible alternative paths to reach D – using node A or node B as next hop. However, routing via node A is not a safe option. This is because from A’s perspective, shortest path to D is via S. Therefore if S sends any packets destined for D towards A, A will route them back to S causing a micro-loop and quickly saturating the bandwidth on the link between the nodes. The safe next hop alternative in this topology is towards node B. When node B receives packets to D it will route it over to C. Therefore Control Plane on node S programs a backup next hop for destination D over interface towards B into the Switching Entity.

In steady state, traffic to D is forwarded only via link to C and the backup next hop is not used. If the link between S and C fails, the Switching Entity immediately switches to forwarding traffic via the protecting next hop. Then, after a while the Control Plane stack converges and new set of next hops is programmed to the Switching Entity.

## Failure detection and switchover triggers

The switchover from the primary next hop to backup can be triggered by both Switching Entity and Control Plane stack, depending on the capabilities of the hardware to detect failures.

If the Switching Entity is not able to run a BFD protocol within the hardware, then it is the Control Plane stack that is first notified of failure and who has to initiate the switchover.

If there is a BFD process running within the hardware, then Switching Entity can perform a switchover without notifying the Control Plane stack when a failure is detected.

Finally, the Switching Entity should also be able to perform a switchover when it detects that the port used by the primary next hop has been brought down.

### Switchover in control plane vs switchover in switching entity

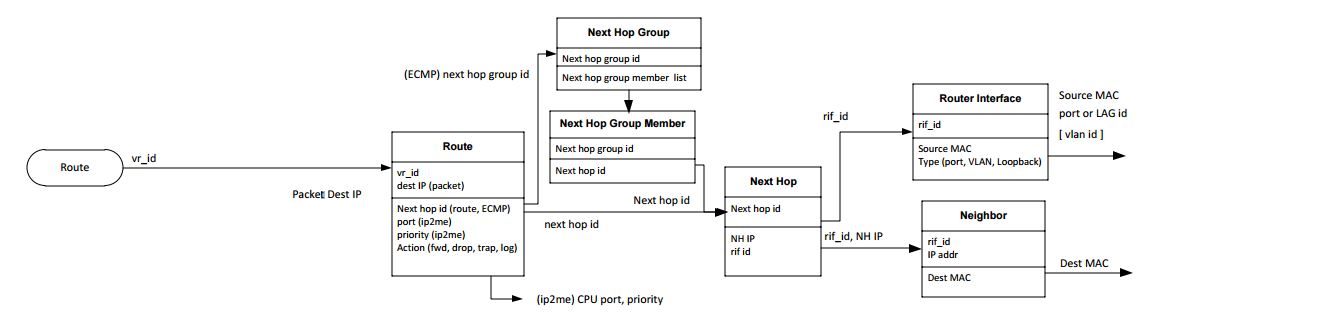
This proposal allows for switchover to be triggered by both control plane and the switching entity.

Not all of the chipsets support BFD in hardware and therefore it is required to support switchovers triggered by the control plane.

At the same time, if the hardware supports BFD, then much faster switchover times can be achieved if the switching entity can trigger them directly. This has been confirmed by feedback we received from hardware vendors after we presented the first draft of this proposal

# Proposal

## Current SAI object model



Current model allows programming multiple next hops for a given destination. This is achieved by using a Next Hop Group object instead of a single Next Hop when specifying Next hop id on the Route object.

Next Hop Group then has a list of Next Hop Group Members each of which points to a single Next Hop.

At the moment this model assumes that the only case when a Next Hop Group is used is ECMP.

## Proposed extensions

In order to allow Fast Reroute programming in the Switching Entity, we propose to extend the Next Hop Group and Next Hop Group Member objects.

### Next Hop Group

Firstly, a new type of Next Hop Group is specified – SAI\_NEXT\_HOP\_GROUP\_TYPE\_PROTECTION. This indicates that the group does not represent ECMP next hops but a primary-backup pair.

It is expected that the Control Plane stack will never attempt to program more than two next hops within a single Protection Next Hop Group. It is outside the scope of this proposal to specify how the Adapter or Adapter Host should enforce this condition.

Furthermore, a new attribute is added to the Next Hop Group to indicate what kind of events can cause the Switching Entity to initiate a switchover. The attribute is SAI\_NEXT\_HOP\_GROUP\_ATTR\_SWITCHOVER\_TYPE and can be any combination of the following bit flags:

* SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_NULL
* SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_BFD
* SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_PORT\_DOWN

Finally, two attributes are added to allow the Control Plane stack to initiate and revert the failover. These are SAI\_NEXT\_HOP\_GROUP\_ATTR\_SET\_SWITCHOVER and SAI\_NEXT\_HOP\_GROUP\_ATTR\_CLEAR\_SWITCHOVER. This is required for example in the scenario when the BFD process runs in the Control Plane process rather than in the Switching Entity and the Control Plane stack has to trigger a switchover “manually”.

### Next Hop Group Member

Two attributes are added to the Next Hop Group Member object. These indicate what is the configured and actual role of the referred next hop in a protection group. The attributes are:

* SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_PREFERRED\_ROLE
  + This attribute is configurable and has to be specified when the next hop group member is created.
  + It can take one of the following values:

|  |  |
| --- | --- |
| SAI\_NEXT\_HOP\_GROUP\_MEMBER\_FORWARDING | The next hop group member is configured as primary and will forward traffic if there is no failure. |
| SAI\_NEXT\_HOP\_GROUP\_MEMBER\_STANDBY | The next hop group member is configured as a backup and will not forward traffic unless the primary fails. |

* SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_OBSERVED\_ROLE
  + This is a read-only attribute which represents the actual role of the referred next hop.
  + It can take one of the following values:

|  |  |
| --- | --- |
| SAI\_NEXT\_HOP\_GROUP\_MEMBER\_FORWARDING | This next hop group member is currently forwarding traffic. Valid for both primary and backup. |
| SAI\_NEXT\_HOP\_GROUP\_MEMBER\_STANDBY | This next hop is backup and is currently not forwarding any traffic. |
| SAI\_NEXT\_HOP\_GROUP\_MEMBER\_FAILED | This next hop is primary but is currently in failed state and is not forwarding any traffic. |

Furthermore, the relationship between next hops and their roles within a protection group can be in one of two states:

|  |  |  |
| --- | --- | --- |
|  | Primary | Backup |
| PREFERRED | FORWARDING | STANDBY |
| OBSERVED | FORWARDING | STANDBY |

|  |  |  |
| --- | --- | --- |
|  | Primary | Backup |
| PREFERRED | FORWARDING | STANDBY |
| OBSERVED | FAILED | FORWARDING |

Failure of primary

Failure cleared

# Specification

## Addition to file sainexthopgroup.h

### Data Structures and Enumerations

#### Changes to Next Hop Group

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

{

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

SAI\_NEXT\_HOP\_GROUP\_TYPE\_ECMP,

**/\*\* Next hop protection group. Contains primary and backup next hops. \*/**

**SAI\_NEXT\_HOP\_GROUP\_TYPE\_PROTECTION,**

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

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

/\*\*

\* @brief Attribute id for next hop

\*/

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

{

/\*\*

\* @brief Start of attributes

\*/

SAI\_NEXT\_HOP\_GROUP\_ATTR\_START,

/\*\*

\* @brief Number of next hops in the group

\*

\* @type sai\_uint32\_t

\* @flags READ\_ONLY

\*/

SAI\_NEXT\_HOP\_GROUP\_ATTR\_NEXT\_HOP\_COUNT = SAI\_NEXT\_HOP\_GROUP\_ATTR\_START,

/\*\*

\* @brief Next hop member list

\*

\* @type sai\_object\_list\_t

\* @flags READ\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_NEXT\_HOP\_GROUP\_MEMBER

\*/

SAI\_NEXT\_HOP\_GROUP\_ATTR\_NEXT\_HOP\_MEMBER\_LIST,

/\*\*

\* @brief Next hop group type

\*

\* @type sai\_next\_hop\_group\_type\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\*/

SAI\_NEXT\_HOP\_GROUP\_ATTR\_TYPE,

**/\*\***

**\* @brief Events that can initiate switchover from primary to backup next hop**

**\***

**\* Can be a combination of the following:**

**\* SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_NULL,**

**\* SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_BFD,**

**\* SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_PORT\_DOWN**

**\***

**\* @type sai\_uint8\_t**

**\* @validonly SAI\_NEXT\_HOP\_GROUP\_ATTR\_TYPE == SAI\_NEXT\_HOP\_GROUP\_TYPE\_PROTECTION**

**\*/**

**SAI\_NEXT\_HOP\_GROUP\_ATTR\_SWITCHOVER\_TYPE,**

**/\*\***

**\* @brief Trigger a switchover from primary to backup next hop**

**\***

**\* @type bool**

**\* @default false**

**\* @validonly SAI\_NEXT\_HOP\_GROUP\_ATTR\_TYPE == SAI\_NEXT\_HOP\_GROUP\_TYPE\_PROTECTION and SAI\_NEXT\_HOP\_GROUP\_ATTR\_CLEAR\_SWITCHOVER == false**

**\*/**

**SAI\_NEXT\_HOP\_GROUP\_ATTR\_SET\_SWITCHOVER,**

**/\*\***

**\* @brief Revert back to forwarding over primary next hop**

**\***

**\* @type bool**

**\* @default false**

**\* @validonly SAI\_NEXT\_HOP\_GROUP\_ATTR\_TYPE == SAI\_NEXT\_HOP\_GROUP\_TYPE\_PROTECTION and SAI\_NEXT\_HOP\_GROUP\_ATTR\_SET\_SWITCHOVER == false**

**\*/**

**SAI\_NEXT\_HOP\_GROUP\_ATTR\_CLEAR\_SWITCHOVER,**

/\*\*

\* @brief End of attributes

\*/

SAI\_NEXT\_HOP\_GROUP\_ATTR\_END,

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

SAI\_NEXT\_HOP\_GROUP\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\*\* End of custom range base \*/

SAI\_NEXT\_HOP\_GROUP\_ATTR\_CUSTOM\_RANGE\_END

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

#### Changes to Next Hop Group Member

**/\*\***

**\* @brief No switchover triggers supported**

**\*/**

**#define SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_NULL 0x00000000**

**/\*\***

**\* @brief Supported switchover triggered by BFD**

**\*/**

**#define SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_BFD 0x00000001**

**/\*\***

**\* @brief Supported switchover triggered by port going down**

**\*/**

**#define SAI\_NEXT\_HOP\_SWITCHOVER\_TRIGGER\_PORT\_DOWN 0x00000002**

**/\*\***

**\* @brief Next hop group member protection role**

**\*/**

**typedef enum \_sai\_next\_hop\_group\_member\_protection\_role\_t**

**{**

**/\*\* Next hop group member is forwarding \*/**

**SAI\_NEXT\_HOP\_GROUP\_MEMBER\_FORWARDING,**

**/\*\* Next hop group member is standby \*/**

**SAI\_NEXT\_HOP\_GROUP\_MEMBER\_STANDBY,**

**/\*\* Next hop group member failed \*/**

**SAI\_NEXT\_HOP\_GROUP\_MEMBER\_FAILED,**

**} sai\_next\_hop\_group\_member\_protection\_role\_t;**

typedef enum \_sai\_next\_hop\_group\_member\_attr\_t

{

/\*\*

\* @brief Start of attributes

\*/

SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_START,

/\*\*

\* @brief Next hop group id

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_NEXT\_HOP\_GROUP

\*/

SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_NEXT\_HOP\_GROUP\_ID = SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_START,

/\*\*

\* @brief Next hop id

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_NEXT\_HOP

\*/

SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_NEXT\_HOP\_ID,

/\*\*

\* @brief Member weights

\*

\* @type sai\_uint32\_t

\* @flags CREATE\_AND\_SET

\* @default 1

\*/

SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_WEIGHT,

**/\*\***

**\* @brief Preferred role in the protection group**

**\***

**\* Should only be used if the type of owning group is SAI\_NEXT\_HOP\_GROUP\_TYPE\_PROTECTION**

**\***

**\* @type sai\_next\_hop\_group\_member\_protection\_role\_t**

**\* @flags CREATE\_ONLY**

**\* @default SAI\_NEXT\_HOP\_GROUP\_MEMBER\_PRIMARY**

**\*/**

**SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_PREFERRED\_ROLE,**

**/\*\***

**\* @brief The actual role in protection group**

**\***

**\* Should only be used if the type of owning group is SAI\_NEXT\_HOP\_GROUP\_TYPE\_PROTECTION**

**\***

**\* @type sai\_next\_hop\_group\_member\_protection\_role\_t**

**\* @flags READ\_ONLY**

**\*/**

**SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_OBSERVED\_ROLE,**

/\*\*

\* @brief End of attributes

\*/

SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_END,

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

SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\*\* End of custom range base \*/

SAI\_NEXT\_HOP\_GROUP\_MEMBER\_ATTR\_CUSTOM\_RANGE\_END

} sai\_next\_hop\_group\_member\_attr\_t;

### API

There are no changes to be made to the API.

# Examples

# Pipeline