

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **SAI Multi NPU Part 2 (Fabric ID)** |
| **Authors** | **Dell** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **7/6/2016** |
| **SAI-Version** | **1.0** |

**Contents**

[List of Changes i](#_Toc457480795)

[1 Overview 2](#_Toc457480796)

[2 Supporting a Multi-NPU networking fabric 4](#_Toc457480797)

[2.1 SAI Unique Fabric/Module ID per NPU 4](#_Toc457480798)

[2.1.1 Specification 4](#_Toc457480799)

[2.2 SAI Remote/Global OID 5](#_Toc457480800)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.3 | Base version |  | 7/8/2016 |
| 1.8.0 | Part 2 contains information related to Fabric ID, used for building a multi NPU networking fabric |  | 8/5/2016 |

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

This proposal is intended to enhance the existing SAI API model in order to support multiple NPU’s, either on multi-card or single-card systems, such as:

 Chassis with control processors (CP) and line cards (LC)

 Stacking

## Acronyms

|  |  |
| --- | --- |
| Network Fabric | Collection of switches connected in Chassis or Stack topologies and present a single logical switch |
| Fabric Leaf Switch | A Switch which contains front panel ports. Traffic from/to front panel ports is connected with Spine switch(es) through fabric ports |
| Fabric Spine Switch | A Switch which routes traffic across fabric ports. This Switch can optionally have front panel ports. |
| Fabric Element | Typically a crossbar switch in a CLOS fabric which inter connects traffic from switches through high-speed serial interconnects. It is possible that these FE devices deal with cells rather than Ethernet packets. |

## Topologies

### Chassis System – with an Spine-Leaf networking fabric (Active Spine)



The “Spine Controller” is realized with a Fabric Card.

### Chassis System – with a Spine-Leaf networking fabric (Passive Spine Element)



Fabric ID’s are not needed for “passive” spine elements.

### Stacking System



# Scope

This section describes how SAI can support two types of fabric: Fabrics with and without Internal Headers.

## Fabric of independent switches

In this architecture, each element in the fabric acts as an independent switch which performs complete ingress and egress functionalities. Ethernet packets are switched in the fabric without any internal headers added to the packets.

In this model, SAI user can consider each switch in the fabric as a separate unit and use exiting SAI objects to design their stack or chassis models to forward packets from one switch to the other switch.

Example:

* Use regular ports as interconnected links with the help of existing SAI objects.
* Lookups shall be done in each switch with forwarding data bases created in all switches.

This architecture may have some limitations as listed below:

- Higher latency because of redundant lookups in each switch along the path in the fabric,

- Each switch will have to repeat QoS checks (Traffic class mapping, drop precedence etc.)

## Fabric with cooperative switches using internal headers

This model covers Architectures where participating switches use proprietary fabric headers to carry meta data needed for forwarding or classifications from ingress port in switch 1 to destination port in other switch 2.

In this model look up will happen only at ingress unit and forwarding will happen based on result of lookup.

* Each switch is interconnected with fabric links
* Packets switching from one switch to other switch will use these fabric links.
* Packets Txing out of fabric ports will be prepended with fabric header (ingress packet data + meta data needed for forwarding).
* Packets Rxing on fabric ports will remove fabric header and will use metadata to forward to egress ports. Info contained in metadata will be used in the egress processing.

This model mandates fabric to be made of a set of cooperative switches with well-defined roles in the ingress, fabric and egress paths.

**We propose to leave management of a passive Fabric Element out of the scope of SAI.**

The rest of the document describes changes required to support a multi-NPU networking fabric.

# Supporting a Multi-NPU networking fabric

The following changes are required to support a multi-NPU networking fabric:

* ssigning a unique ID to each NPU in a networking fabric (Fabric ID).
* Fabric ports and behavior.
* Remote port (global port identifier in a networking fabric)
* SAI object creation by providing HW ID - needed for LAG and MGID
  + In a multi-NPU networking fabric, multi-NPU LAG / trunk groups must be uniquely identified by a global LAG ID; same concept applies to other SAI objects
* Fabric forwarding table management.

## SAI Unique Fabric ID per NPU

In Chassis or stacking systems each NPU in system needs the unique identifier for fast path switching. By looking at this identifier packets are switched to destination NPU.

For instance, the fabric ID can be used to distinguish port 1 on switch x from port 1 on switch y; essentially, each individual port is globally identified by the pair [Fabric ID, Local NPU Port ID].

Use Cases:

* Unicast FDB – Can point to the port in other NPU.
* L3 Routing - Destination port can be other NPU.

This is different from a switch ID whose scope is local to the controlling CPU. This Fabric ID scope is global to chassis or stack system.

Note. It is assumed “fabric ID” is a platform specific parameter, managed by the Host Adapter. In fact, determining “fabric ID’s” may be the responsibility of higher software layers, but this is outside the scope of the SAI API specification.

**Topology Discovery:**

General chassis or stacking model will have their own topology discovery modules to discover participating NPUs in the system and how they are connected each other. After discovering topology unique fabric/module IDs will be assigned to each NPU in the system. The mechanisms used for discovery of topology is outside the scope of SAI.

**Dynamic Discovery:**

- Every switch will participate in discovery with existing fabric id. Example Default fabric id 0 for all units.

- Once Topology discovered, each unit will be assigned with a unique fabric ID in that system.

**Fixed Topology:**

- In Fixed Chassis, User can predefine fabric id’s for each switch in their product.

SAI should have option to set the fabric id as part of switch initialization or after switch init. This depends on the SWITCH capability.

## Fabric Connections

Switches participating in a fabric are inter connected through fabric links. In a Clos fabric, front panel switches are connected to fabric elements, as show in earlier diagram.

### Fabric Ports

In a stacking scenario, a set of ports in NPU will be used to connect as fabric links or backplane. These ports may send the packets from one unit to other unit with proprietary encapsulation. This is transparent to the SAI API (except for setting the encapsulation protocol, as a vendor extension)

We propose to provide the option of setting ports in fabric mode. Based on platform details, Adaptor host will configure a set of ports in fabric mode.

Fabric ports can be part of regular switch or fabric elements. For Fabric port VLAN membership and STP states checks are not needed. SAI should make sure this behavior when port set as fabric port. This will make sure, model is same in case of fabric port is part of FE or regular switch.

All physical port attributes are valid for fabric port also. Fabric port can be used to create the Fabric LAG and bridge port creation in order to use this as part of Multicast group id etc.

### Encapsulation Protocol

Fabric encapsulation protocol is considered as proprietary to a Hardware family. It is also possible that within a HW family, there are multiple versions for these encapsulations. SAI should allow configuration of these vendor specific encapsulation types, using SAI vendor extensions (thus outside the scope of the document). The Host adapter should ensure that compatible versions are used in a given fabric. The logical pipe line model, proposed by SAI, will however define the contents of metadata carried in the fabric.

### Fabric Header

Fabric header will be appended to the original packet, below are logical header fields

* In\_port
* Out\_port – Physical or Multicast group id.
* Out \_if
* TC and DP

Note that this does not impact the SAI API specification.

## Multi NPU General Pipe Line



## SAI OID

In a Multi-NPU system, host-adaptor will hold OIDs generated by the SAI instances.

It is required that SAI OIDs used for forwarding are unique in the fabric.

As SAI OIDs are internal to a SAI implementation, Inter-op between different SAI implementations is not expected.

## SAI Port

In a Multi-NPU system, global or system-wide NPU ports refer to the union of all ports present in all participating NPUs. These NPUs can be present in a different line cards for a chassis system OR present in the units of a stacking system.

As mentioned above, Egress port of forwarding tables can be a local port or a remote port. In order to support this every SAI instance participating in a multi-NPU system should also be aware of the ports present in other NPUs.

**Local Port:**  port is physically present in the same switch.

**Remote Port:** Port is physically present in a remote switch which is connected through the fabric.

Note that a port OID can represent both local port and remote port.

Use cases:

SAI APIs permit MAC or Route table to be configured with a global port. If the destination port is a remote port, then the packet is transmitted (directed) through the fabric towards the remote NPU.

SAI OID’s between SAI instances:

Host-adaptor is expected to configure each NPU in the system with its local ports as well as remote ports belonging to other NPUs in the system.

Remote port OIDs, cached in each SAI instance, will be used to deduce fabric id of the remote NPU and hardware port id in that NPU.

**Configuration Example:**

* Line card1 - switch 0, Port 0(L1-P0), Fabric ID – 103 and Controlled by host-adaptor 0 in LC 1.
* Line card2 - switch 0, Port 0(L2-P0), Fabric ID – 105 and Controlled by host-adaptor 1 in LC 2.
* Spine 0 : Connections
  + LC 1 switch 0 Fabric port 0(L1-FP0) to spine fabric port 0(SFP-0).
  + LC 2 switch 0 Fabric port 0(L2-FP0) to spine fabric port 1(SFP-1).

**Host-adaptor 0:**

* Retrieve the port oid list from switch 0.
  + Port 0 oid = 100
* Retrieve port’s fabric id and hardware port id.
  + Fabric ID = 103
  + Local port number = 0.
* Learn mac address in port 0 i.e

Mac = 00:00:00:11:11:11 Vlan=100 Port =100 (oid of switch 0, port 0 i.e 100).

**Host-adaptor 1:**

* Retrieve the port oid list from switch 0 linecard 1.
  + Port 0 oid = 100
* Retrieve port fabric id and hardware port id for 100.
  + Fabric ID = 105
  + Local port number = 0.
* Program learned MAC address in switch 0 in Line card 0 in switch 0 in Line card 1. To do this we need the information of port oid 100.
  + Adaptor host 1 needs to sync to port oid in adaptor host 0 and get the Fabric and local port number of port oid 100.
  + Create port in switch 0 line card 1 bypassing the fabric + local port information (103, 0). This will give the new port oid(remote port oid) i.e 200.
  + Use this oid for programming the FDB entry.

Mac = 00:00:00:11:11:11 Vlan=100 Port = 200.

### SAI VLAN

Use case, nothing to do with SAI API

Programming each NPU in the system with VLAN membership for all participating ports (local and remote) will pose scaling issues. Hence it is proposed that each NPU is only programmed with its local VLAN membership.

It mandates that ingress VLAN checks are performed in the ingress NPU and egress VLAN checks are performed in the egress NPU.

Note that fabric ports are also considered as local ports of the NPU for this discussion.

Configuration Example:

* Refer example above
* L1P-0 and L12P-0 are members of vlan 100.
* Flood traffic:
  + Ingress traffic on L1P-0 , to Flood traffic to LC2 it should go L1FP-0 and L2FP-0.
  + Add L1FP-0 as member of VLAN 100 in LC1.
  + Add L2FP-0 as member of VLAN 100 in LC2.

### SAI LAG

In chassis or stacking architectures, LAG member ports can span across multiple switches. In SAI pipe line, hash resolution will happen at ingress NPU and the resultant destination port can be either a local port or a remote port. Otherwise, packet needs to be sent to all NPUs which are part of the LAG span making the design sub-optimal in terms of fabric bandwidth used.

This model proposes that each NPU is configured with all member ports of the LAG.

### SAI STP

Ingress STP checks will happen at ingress switch and egress destination checks will happen at the destination switch. Scalability is again the rationale behind this approach. Populating stp database for all ports (local/remote) in NPU can be a challenge in systems with higher port density.

That apart, destination ports for BUM traffic will be identified only at egress NPUs.

Hence this model proposes that STP checks are performed at the egress NPUs alone.

However applications have the option of flushing FDB entries when a port goes into blocking state. This way fabric bandwidth can be saved.

### SAI FDB

In the proposed model, FDB entry will be leaned at each ingress NPU and same will be programmed in all other NPUs with the destination as port where packets received. If the result of lookup indicates that the destination port is remote, then HW will use fabric route table to determine that fabric ports to be used. Before sending the packet on fabric ports, internal meta data will be updated with the results of the lookup. Once the egress NPU receives the packet, it used the internal meta data to identify the destination port. STP and VLAN membership checks are performed before sending the packet out of the destination port.

### SAI L3

As per SAI pipeline model, Route table points to a LAG or a physical port. If the destination port is a remote port, then Fabric routing table is used to identify the fabric port(s) to be used for internal routing. This model mandates that packet modifications will happen at ingress unit.

### SAI Multicast

TBD

## SAI Fabric Forwarding Table

In general, in a Chassis or Stacking environment, routing traffic from one NPU to other NPU is managed

by a Fabric forwarding table based on fabric id’s and fabric ports.

This table helps to:

- Obtain the shortest path to reach the destination NPU.

- Preventing loops.

This table can be changed based on topology changes and this topology changes can be detected from software or hardware. Controlling this table form host depends on NPU/SDK capability.

**Example:** Refer Figure 3.

To reach fabric switch 103 from Fabric switch 101, there is 2 fabric ports connected. Based on topology discover, adaptor host can configure

Fabric ID 103 can reach by Fabric port directly connected to 101.

## SAI Object Creation By passing ID

There is a requirement to have the notion of global Port OID as captured in the Section 4 SAI Remote/Global OID. Further, there is also a need for having global OID for certain SAI objects in the Multi NPU system. This is required when the forwarding lookup happens only in ingress NPU and the destination information is carried as meta-data to the egress NPU. Some examples are captured below.

**Examples: LAG object**

 SAI LAG object can have member ports spanning across multiple NPU in the system. If a packet

enters the ingress NPU on one of the LAG member ports, the Source LAG Id assigned for the packet is usually carried as a meta-data to the remote NPU for reasons like flooding the packets to member ports in the remote NPU.

 This requires the LAG Id to be globally unique across the multiple NPUs.

Multicast Group ID

 When a packet is destined to a Multicast Group Id that has member ports in remote NPU, the Multicast Group Id has to be looked up in the remote NPU for replicating to the local egress ports that are part of the group.

 This requires the multicast group id to be globally unique across the multiple NPUs.

To provide the more flexibility and control to host, SAI API needs to be extended to program object with the given Id. This will also solve the Multi NPU requirements.

# Specification

## Switch

### Specification

This section describes an overview of the proposed API changes in order to support a fabric ID.

/\* Set Fabric ID allowed only after deleting all SAI objects from switch\*/ SAI\_SWITCH\_ATTR\_FABRIC\_ID

#define SAI\_KEY\_SWITCH\_FABRIC\_ID "SAI\_SWITCH\_FABRIC\_ID"

## Port

{

/\*\* Fabric Port \*/

SAI\_PORT\_TYPE\_FABRIC,

} sai\_port\_type\_t;

/\* Faric Id of port \*/

SAI \_PORT\_ATTR\_FABRIC\_ID,

/\* Local hardware port id in switch \*/

SAI\_PORT\_ATTR\_HW\_ID,

A new port attribute is added, to specify the list of fabric ID’s that can be accessed through the given

port.

/\*\* Fabric ID list [uint8\_t]

\* Port can be used to route traffic to fabric in the list.

\* Modify needs to provide the new list.

\* Only Valid for fabric port oid’s.

\*/

SAI\_PORT\_ATTR\_FABRICPORT\_TO\_FABRIC\_ID\_MAPPING,

## With ID

Proposal is to have the attribute in all the objects as platform specific and opaque (to the host adapter)

“hardware identifier”. This identifier needs to be provided at object creation time.

As explained in pipeline, MGID is carried as metadata from one switch to other switch and refer multicast data base in every switch and replicate packets out\_if list. In this case MGID should be similer in all the units. In CURD model to we can achieve this by only passing input and get similar output from all SAI instances.

**Example:**

/\*\* SAI LAG HWID list [sai\_u32\_list\_t] \*/ SAI\_LAG\_ATTR\_HWID\_LIST,