

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **Subport interface , .1D Bridges , bridge port interface** |
| **Authors** | **Mellanox** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **07/18/2016** |
| **SAI-Version** | **1.0** |

1 Overview 3

2 Sub port interface 3

2.1 New port type attribute 3

2.2 Exmple - Router sub port flow 4

 4

3 Non Vlan aware bridge domains (.1D bridges) 4

3.1 Non Vlan aware bridge flows 5

4 Bridge ports 6

4.1 SAI bridge/router object current state 6

4.2 SAI bridge/router object proposed state 7

4.3 Bridge port type types 7

4.3.1 Bridge port type - Port 8

4.3.2 Bridge vport type – { Port , vlan} 8

4.3.3 Bridge port type- router 8

4.3.4 Bridge port type – tunnel 9

4.4 Bridge attributes 10

5 Specification 10

5.1 SAI port object 10

5.2 SAI FDB object 10

5.3 SAI STP object 10

5.4 SAI router interface object 10

5.5 SAI bridge port object 11

5.5.1 Bridge port API summary 12

5.6 SAI bridge object 12

5.6.1 Bridge API summary 13

6 Examples 13

7 References 13

# Overview

The proposal contain three prats

* In order build a discrete pipeline and well define layers additional set of object are needed Bridge ports, Currently when entering the switch we have -> ingress port /LAG When entering a router we have SAI RIF A bridge port object is missing
* increase SAI 4k vlan broadcast domain by adding multiple vlan unaware bridges (.1D) to te existing Vlan aware bridge (1.Q)
* add an ability to define an interface base on {port,vlan} were interface can be router interface or bridge interface

# Sub port interface

Ability to define an interface base on {port,vlan} were interface can be router interface or bridge interface



## New port type attribute

Control on the port mode optional modes

* Port – legacy
* Sub,port- port is splitter into logical interfaces each interface is define by {port,vlan}

## Exmple - Router sub port flow

# 

# Non Vlan aware bridge domains (.1D bridges)

Increase SAI 4k vlan broadcast domain by adding multiple vlan unaware bridges (.1D) to the existing Vlan aware bridge (1.Q)

Add ability to create .1D bride object

Add ability to bind an "interface" to the a .1D bridge

* only interface type {port,vlan}



## Non Vlan aware bridge flows



# Bridge ports

In order build a discrete pipeline and well define layers additional set of object are needed Bridge ports, Currently when entering the switch we have -> ingress port /LAG When entering a router we have SAI RIF A bridge port object is missing

## SAI bridge/router object current state



## SAI bridge/router object proposed state



## Bridge port type types

Port – represent phy port or LAG traditional .1Q bridge port

Vport - represent phy port or LAG. vlan .1Q bridge port interface

Router - represent the port that connect the bridge to the router

Tunnel - represent the port that connect the bridge to tunnel



.

### Bridge port type - Port

Represent phy port or LAG traditional .1Q bridge port

Can be added into .1Q Bridge only

Can be binded only form a port mode {port}

By default all port will have a corresponding bride port add to the default 1.q bridge

Valid attribute:



### Bridge vport type – { Port , vlan}

Represent phy port or LAG .vlan

Can be added into .1D Bridge only

Can be binded only form a port mode {vport}



### Bridge port type- router

Represent to bridge port connect to the router

Two types of router bridge port

1.Q router bridge port- a single port representing all router Vlan interfaces

Per bridge 1.D router bridge port per 1.D bridge port



### Bridge port type – tunnel

Represent to bridge port connect to a tunnel

There is one to one mapping between the bride port and the tunnel



## Bridge attributes

All bridge related attribute will move from port/LAG to bridge port object

Bridge attribute

STP

Vlan objects

…

# Specification

## SAI port object

**typedef enum \_sai\_port\_bind\_attr\_t**

**{**

**SAI\_PORT\_MODE,**

**SAI\_SUB\_PORT\_MODE,**

**} sai\_port\_\_bind\_attr\_t ;**

typedef enum \_sai\_port\_attr\_t

{

//new attribute

/\*\* port type[sai\_port\_bind\_attr\_t] \*/

**SAI\_PORT\_BIND\_MODE,**

//removed atrtributes

SAI\_PORT\_ATTR\_FDB\_LEARNING,

SAI\_PORT\_ATTR\_MAX\_LEARNED\_ADDRESSES,

SAI\_PORT\_ATTR\_FDB\_LEARNING\_LIMIT\_VIOLATION ,

} sai\_port\_attr\_t ;

## SAI FDB object

typedef struct \_sai\_fdb\_entry\_t

{

sai\_mac\_t mac\_address;

**sai\_fdb\_entry\_br\_type br\_type; // .1D or .1Q**

**//valid for .1Q**

**sai\_vlan\_id\_t vlan\_id;**

**//valid for .1D**

**sai\_bridge\_id\_t br\_id;**

} sai\_fdb\_entry\_t;

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

{

//new attribute

SAI\_FDB\_ENTRY\_ATTR\_BRIDGE\_PORT\_ID

//removed attributes

SAI\_FDB\_ENTRY\_ATTR\_PORT\_ID

}

## SAI STP object

Set set port state- Panding STP port state – change to object

The port state should be move from port to bridge port

typedef enum \_sai\_stp\_attr\_t

{

SAI\_STP\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* Vlans attached to STP instance [sai\_vlan\_list\_t] \*/

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

//new attribute

/\*\* bridge attached to STP instance \*/

SAI\_STP\_ATTR\_BRIDGE\_ID

/\*\* READ-WRITE \*/

SAI\_STP\_ATTR\_END,

} sai\_stp\_attr\_t;

## SAI VLAN member object

typedef enum \_sai\_vlan\_member\_attr\_t {

SAI\_VLAN\_MEMBER\_ATTR\_START,

/\*\* READ\_WRITE \*/

/\*\* VLAN ID [sai\_vlan\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

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

/\*\* logical port ID [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_VLAN\_MEMBER\_ATTR\_PORT\_ID,

**/\*\* logical bridge port ID valid only for .1Q bridge ports [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/**

**SAI\_VLAN\_MEMBER\_ATTR\_BRIDGE\_PORT\_ID,**

/\*\* VLAN tagging mode [sai\_vlan\_tagging\_mode\_t] (CREATE\_AND\_SET)

\* (default to SAI\_VLAN\_PORT\_UNTAGGED) \*/

SAI\_VLAN\_MEMBER\_ATTR\_TAGGING\_MODE,

SAI\_VLAN\_MEMBER\_ATTR\_END,

/\*\* custom range base value \*/

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

/\* --\*/

SAI\_VLAN\_MEMBER\_ATTR\_CUSTOM\_RANGE\_END

} sai\_vlan\_member\_attr\_t;

## SAI router interface object

/\*\*

\* @brief Attribute data for SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE

\*/

typedef enum \_sai\_router\_interface\_type\_t

{

/\*\* Port or Lag Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_PORT,

/\*\* VLAN Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_VLAN,

/\*\* Loopback Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK

**/\*\* sub port Interface Type \*/**

**SAI\_ROUTER\_INTERFACE\_TYPE\_SUB\_PORT**

**/\*\* .1D bridge Interface Type \*/**

**SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE**

} sai\_router\_interface\_type\_t;

typedef enum \_sai\_router\_interface\_attr\_t

{

/\*\* READ-ONLY \*/

/\*\* Virtual router id [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID,

/\*\* Type [sai\_router\_interface\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE,

/\*\* Assosiated Port or Lag object id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_PORT or SUB\_PORT | CREATE\_ONLY)

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_PORT\_ID,

/\*\* Assosiated Vlan [sai\_vlan\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_VLAN or SUB\_OPORT | CREATE\_ONLY)

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_VLAN\_ID,

/\*\* Assosiated Vlan [sai\_bridge\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE | CREATE\_ONLY)

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_BR\_ID,

## SAI bridge port object

/\*\*

\* @brief Attribute data for SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE

\*/

typedef enum \_sai\_bridge\_port\_type\_t

{

/\*\* Port or Lag \*/

SAI\_BRIDGE\_PORT\_TYPE\_PORT,

/\*\* {Port or Lag.vlan} \*/

SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT,

/\*\* bridge router port \*/

SAI\_BRIDGE\_PORT\_TYPE\_1Q\_ROUTER,

/\*\* bridge router port \*/

SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER,

/\*\* bridge tunnel port \*/

SAI\_BRIDGE\_PORT\_TYPE\_TUNNEL,

} sai\_bridge\_port\_type\_t;

typedef enum \_sai\_bridge\_port\_attr\_t

{

/\*\* Type [sai\_bridge\_port\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_BRIDEG\_PORT\_ATTR\_TYPE,

/\*\* Assosiated Port or Lag object id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_PORT or SAI\_BRIDGE\_PORT\_TYPE\_PORT | CREATE\_ONLY)

\*/

SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID,

/\*\* Assosiated vlna object id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_PORT | CREATE\_ONLY)

\*/

SAI\_BRIDGE\_PORT\_ATTR\_VLAN\_ID,

/\*\* Assosiated tunel id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_TUNNEL | CREATE\_ONLY)

\*/

SAI\_BRIDGE\_PORT\_ATTR\_TUNNEL\_ID,

/\*\* Assosiated tunel id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER | CREATE\_ONLY)

\*/

SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID,

//bridge port attribute

SAI\_BRIDGE\_PORT\_ATTR\_FDB\_LEARNING,

SAI\_BRIDGE\_PORT\_ATTR\_MAX\_LEARNED\_ADDRESSES,

SAI\_BRIDGE\_PORT\_ATTR\_FDB\_LEARNING\_LIMIT\_VIOLATION ,

} sai\_bridge\_port\_attr\_t ;

/\*\*

\* @brief Create bridge port

\*

\* @param[out] bridge\_port\_id

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

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

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_bridge\_port\_fn)(

\_Out\_ sai\_object\_id\_t\* bridge\_port\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* @brief Remove bridge port

\*

\* @param[in] bridge\_port\_id.

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_brideg\_port\_fn) (

\_In\_ sai\_object\_id\_t brdge\_port\_id

);

/\*\*

\* @brief Set attributes for bridge port

\*

\* @param[in] bridge port id

\* @param[in] attr attribute to set

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_bridge\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_port\_id,

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

);

/\*\*

\* @brief Get attrbutes of bridge port

\*

\* @param[in] bridge\_port\_id

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

\* @param[inout] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_bridge\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_port\_id ,

\_In\_ uint32\_t attr\_count,

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

);

### Bridge port API summary

typedef struct bridge\_port\_t

{

sai\_create\_bridge\_port\_fn create\_bridge\_port;

sai\_remove\_bridge\_port\_fn remove\_bridge\_port;

sai\_set\_bridge\_port\_attribute\_fn set\_bridge\_port\_attribute;

sai\_get\_bridge\_port\_attribute\_fn get\_bridge\_port\_attribute;

} sai\_bridge\_port\_api\_t;

## SAI bridge object

typedef enum \_sai\_bridge\_attr\_t

{

/\*\* List of bridge port members [sai\_object\_list\_t]\*/

SAI\_BRIDGE\_ATTR\_BRIDE\_PORT\_LIST,

} sai\_bridge\_attr\_t ;

/\*\*

\* @brief Create bridge

\*

\* @param[out] bridge\_port\_id

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

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

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_bridge\_fn)(

\_Out\_ sai\_object\_id\_t\* bridge\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* @brief Remove bridge

\*

\* @param[in] bridge\_id.

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_brideg\_fn) (

\_In\_ sai\_object\_id\_t brdge\_id

);

/\*\*

\* @brief Set attributes for bridge

\*

\* @param[in] bridge id

\* @param[in] attr attribute to set

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_bridge\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_id,

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

);

/\*\*

\* @brief Get attrbutes of bridge

\*

\* @param[in] bridge\_id bridge\_id

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

\* @param[inout] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_bridge\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_id ,

\_In\_ uint32\_t attr\_count,

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

);

### Bridge API summary

typedef struct bridge\_t

{

sai\_create\_bridge\_fn create\_bridge;

sai\_remove\_bridge\_fn remove\_bridge;

sai\_set\_bridge\_attribute\_fn set\_bridge\_attribute;

sai\_get\_bridge\_attribute\_fn get\_bridge\_attribute;

} sai\_bridge\_api\_t;

# Examples

TBD add an example

## 1.D bridge example

TBD

## Router sub port example

TBD

# References