

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **Uniform tunnel API’s** |
| **Authors** | **Mellanox , metaSwitch** |
| **Status** | **First draft** |
| **Type** | **Standards Track** |
| **Created** | **08/01/2015** |
| **SAI-Version** | **V0.9.4** |

**Contents**

[List of Changes i](#_Toc426209702)

[1 Overview 1](#_Toc426209703)

[1.1 Uniform tunnel model 1](#_Toc426209704)

[1.2 IPinIP tunnel 1](#_Toc426209705)

[1.2.1 IPinIP encap 1](#_Toc426209706)

[1.2.2 IPinIP decap 2](#_Toc426209707)

[1.3 Vxlan tunnel 2](#_Toc426209708)

[1.3.1 Vxlan encap 2](#_Toc426209709)

[1.3.2 Vxlan decap 3](#_Toc426209710)

[1.4 MPLS tunnel 3](#_Toc426209711)

[1.4.1 Ingres LER (MPLS encap) 3](#_Toc426209712)

[1.4.2 LSR enacp 4](#_Toc426209713)

[1.4.3 Egress LER (MPLS decap) 4](#_Toc426209714)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.4 | Proposal for uniform tunnel |  | 8/1/15 |

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

## Uniform tunnel object



## Uniform tunnel encap

## 

## IPinIP tunnel

### IPinIP encap



### IPinIP decap



## Vxlan tunnel

### Vxlan encap



### Vxlan decap

### 



## MPLS tunnel

### MPLS encap



### MPLS decap



# Specification

New object needed:

Next hop:

Tunnel decap,

Tunnel encap,

MPLS next-hop(NHLFE)

Generic Tunnel object

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 |
| /\*\* VLAN Router Interface Type \*/ |
| SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK |

} sai\_router\_interface\_type\_t

## SAI types

typedef struct \_sai\_tunnel\_map\_params\_t

{

/\*\* ECN \*/

sai\_uint8\_t ecn;

/\*\* vlan id \*/

sai\_uint16\_t vlan\_id;

/\*\* VNI id \*/

sai\_uint32\_t vni\_id;

} sai\_tunnel\_map\_params\_t;

typedef struct \_sai\_tunnel\_map\_t

{

/\*\* Input parameters to match \*/

sai\_tunnel\_map\_params\_t key;

/\*\* Output map parameters \*/

sai\_tunnel\_map\_params\_t value;

} sai\_tunnel\_map\_t;

typedef struct \_sai\_tunnel\_map\_list\_t

{

/\*\* Number of entries in the map \*/

uint32\_t count;

/\*\* Map list \*/

sai\_tunnel\_map\_t \* list;

} sai\_tunnel\_map\_list\_t;

## FDB

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

{

/\*\* 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,

} sai\_fdb\_entry\_attr\_t;

## Next hop

/\*\*

\* @brief Next hop type

\*/

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

{

SAI\_NEXT\_HOP\_IP,

/\*MPLS(NHLFE) next hop \*/

SAI\_NEXT\_HOP\_MPLS,

/\*tunnel next hop \*/

SAI\_NEXT\_HOP\_TUNNEL\_DECAP,

SAI\_NEXT\_HOP\_TUNNEL\_ENCAP,

} sai\_next\_hop\_type\_t;

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

{

/\*\* READ-ONLY \*/

/\*\* READ-WRITE \*/

/\*\* Next hop entry type [sai\_next\_hop\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_TYPE,

/\*\* Next hop entry ipv4 address [sai\_ip\_address\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_NEXT\_HOP\_ATTR\_TYPE = SAI\_NEXT\_HOP\_IP)

\* (CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_IP,

/\*\* Next hop entry router interface id [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_ROUTER\_INTERFACE\_ID,

/\* -- \*/

/\*\* Next hop entry tunnel-dst [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_NEXT\_HOP\_ATTR\_TYPE=SAI\_NEXT\_HOP\_TUNNEL\_ENCAP)

\* (CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_DST,

/\*\* Next hop entry tunnel-id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_NEXT\_HOP\_ATTR\_TYPE = SAI\_NEXT\_HOP\_TUNNEL\_DECAP | SAI\_NEXT\_HOP\_TUNNEL\_ENCAP)

\* (CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_ID,

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

SAI\_NEXT\_HOP\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_next\_hop\_attr\_t;

} sai\_next\_hop\_api\_t;

## Tunnel mapper

\*/

/\*\*

\* @brief Enum defining tunnel map types.

\*/

typedef enum \_sai\_tunnel\_map\_type\_t

{

/\*\* TUNNEL Map overlay ECN to underlay ECN \*/

SAI\_TUNNEL\_MAP\_OECN\_TO\_UECN = 0x00000001,

/\*\* TUNNEL Map underlay ECN to overlay ECN \*/

SAI\_TUNNEL\_MAP\_UECN\_TO\_OECN = 0x00000002,

/\*\* TUNNEL Map VNI to VLAN ID \*/

SAI\_TUNNEL\_MAP\_VNI\_TO\_VLAN\_ID = 0x00000003,

/\*\* TUNNEL Map VLAN ID to VNI \*/

SAI\_TUNNEL\_MAP\_VNI\_TO\_VLAN\_ID = 0x00000004,

/\* -- \*/

/\* Custom range base value \*/

SAI\_TUNNEL\_MAP\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_tunnel\_map\_type\_t

typedef enum \_sai\_tunnel\_map\_attr\_t

{

/\*\* tunnel Map type [sai\_tunnel\_map\_type\_t](MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_TUNNEL\_MAP\_ATTR\_TYPE = 0x00000000,

/\*\* tunnel mapper [sai\_tunnel\_map\_list\_t],

\*/

SAI\_TUNNEL\_MAP\_ATTR\_MAP\_TO\_VALUE\_LIST = 0x00000001,

/\* -- \*/

/\* Custom range base value \*/

SAI\_TUNNEL\_MAP\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_tunnel\_map\_attr\_t ;

/\*\*

\* @brief Create tunnel Map

\*

\* @param[out] tunnel\_map\_id tunnel Map 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\_tunnel\_map\_fn)(

\_Out\_ sai\_object\_id\_t\* tunnel\_map\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* @brief Remove tunnel Map

\*

\* @param[in] tunnel\_map\_id tunnel Map id to be removed.

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_tunnel\_map\_fn) (

\_In\_ sai\_object\_id\_t tunnel\_map\_id

);

/\*\*

\* @brief Set attributes for tunnel map

\*

\* @param[in] tunnel\_map\_id tunnel Map Id

\* @param[in] attr attribute to set

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_tunnel\_map\_attribute\_fn)(

\_In\_ sai\_object\_id\_t tunnel\_map\_id,

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

);

/\*\*

\* @brief Get attrbutes of tunnel map

\*

\* @param[in] tunnel\_map\_id tunnel map 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\_tunnel\_map\_attribute\_fn)(

\_In\_ sai\_object\_id\_t tunnel\_map\_id ,

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* tunnel Map methods table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_tunnel\_map\_api\_t

{

sai\_create\_tunnel\_map\_fn create\_tunnel\_map;

sai\_tunnel\_tunnel\_map\_fn remove\_tunnel\_map;

sai\_set\_tunnel\_map\_attribute\_fn set\_tunnel\_map\_attribute;

sai\_get\_tunnel\_map\_attribute\_fn get\_tunnel\_map\_attribute;

}sai\_tunnel\_map\_api\_t;

## Tunnel object

typedef enum \_sai\_tunnel\_type\_t

{

SAI\_TUNNEL\_IPINIP,

SAI\_TUNNEL\_IPINIP\_GRE,

SAI\_TUNNEL\_IPINIP\_GRE\_AND\_KEY

SAI\_TUNNEL\_VXLAN,

SAI\_TUNNEL\_MPLS,

…

} sai\_tunnel\_type\_t;

typedef enum \_sai\_tunnel\_ttl\_mode\_t

{

SAI\_TUNNEL\_TTL\_COPY\_FROM\_INNER,

SAI\_TUNNEL\_TTL\_USER\_DEFINE

} sai\_tunnel\_ttl\_mode\_t

typedef enum \_sai\_tunnel\_dscp\_mode\_t

{

SAI\_TUNNEL\_DSCP\_COPY\_FROM\_INNER,

SAI\_TUNNEL\_DSCP\_USER\_DEFINE

} sai\_tunnel\_dscp\_mode\_t

typedef enum \_sai\_tunnel\_encap\_t

{

/\*\* READ-WRITE \*/

/\*\* tunnel ip verssion ipv4/ipv6 (MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_ATTR\_TYPE=SAI\_TUNNEL\_IPINIP,SAI\_TUNNEL\_IPINIP\_GRE,SAI\_TUNNEL\_IPINIP\_GRE\_AND\_KEY)

(CREATE\_ONLY) \*/

SAI\_TUNNEL\_IP\_VER,

/\*\* tunnel src ip (MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_ATTR\_TYPE=SAI\_TUNNEL\_IPINIP,SAI\_TUNNEL\_IPINIP\_GRE,SAI\_TUNNEL\_IPINIP\_GRE\_AND\_KEY)

(CREATE\_ONLY) \*/

SAI\_TUNNEL\_SRC\_IP,

/\*\* tunnel TTL mode (copy from inner or user define [sai\_tunnel\_ttl\_mode\_t] MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_ATTR\_TYPE=SAI\_TUNNEL\_IPINIP,SAI\_TUNNEL\_IPINIP\_GRE,SAI\_TUNNEL\_IPINIP\_GRE\_AND\_KEY)

(CREATE\_ONLY) \*/

SAI\_TUNNEL\_TTL\_MODE,

/\*\* tunnel TTL value MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_TTL\_MODE = SAI\_TUNNEL\_TTL\_USER\_DEFINE)

SAI\_TUNNEL\_TTL\_VAL,

/\*\* tunnel dscp mode (pipe or uniform model ) [sai\_tunnel\_dscp\_mode\_t] MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_ATTR\_TYPE=SAI\_TUNNEL\_IPINIP,SAI\_TUNNEL\_IPINIP\_GRE,SAI\_TUNNEL\_IPINIP\_GRE\_AND\_KEY)

(CREATE\_ONLY) \*/

SAI\_TUNNEL\_DSCP\_MODE,

/\*\* tunnel DSCP value MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_DSCP\_MODE = SAI\_TUNNEL\_DSCP\_USER\_DEFINE)

SAI\_TUNNEL\_DSCP\_VAL,

/\*\* tunnel GEP key (MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_ATTR\_TYPE=SAI\_TUNNEL\_IPINIP\_GRE\_AND\_KEY)

(CREATE\_ONLY) \*/

\*/

SAI\_TUNNEL\_GRE\_KEY,

/\*\* tunnel mapper [sai\_object\_id\_t] \*/

SAI\_TUNNEL\_ENCAP\_MAPPER,

} sai\_tunnel\_encap\_t;

typedef enum \_sai\_tunnel\_decap\_t

{

/\*\* READ-WRITE \*/

/\*\* enable decap src ip validation check MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_ATTR\_TYPE=SAI\_TUNNEL\_IPINIP,SAI\_TUNNEL\_IPINIP\_GRE,SAI\_TUNNEL\_IPINIP\_GRE\_AND\_KEY)

\*/

SAI\_TUNNEL\_DECAP\_SIP\_CHECK,

/\*\* expected tunnel src ip (MANDATORY\_ON\_CREATE when SAI\_TUNNEL\_DECAP\_SIP\_CHECK is enabled)

(CREATE\_ONLY) \*/

SAI\_TUNNEL\_EXPECTED\_SRC\_IP,

/\*\* tunnel decap ECN mapping [sai\_object\_id\_t] \*/

SAI\_TUNNEL\_DECAP\_ECN\_TABLE,

/\*\* tunnel mapper [sai\_object\_id\_t] \*/

SAI\_TUNNEL\_DECAP\_MAPPER,

} sai\_tunnel\_decap\_t;

typedef enum \_sai\_tunnel\_id\_t

{

/\*\* READ-WRITE \*/

/\*\* tunnel type [sai\_tunnel\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_TUNNEL\_ATTR\_TYPE;

/\*\* tunnel underlay interface [sai\_object\_id\_t] \*/

SAI\_TUNNEL\_UNDERLAY\_INTERFACE;

/\*\* tunnel overlay interafce [sai\_object\_id\_t] \*/

SAI\_TUNNEL\_OVERLAY\_INTERFACE;

/\*\* tunnel encap attribute [sai\_tunnel\_encap\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_TUNNEL\_ENCAP\_ATTR;

/\*\* tunnel dencap attribute [sai\_tunnel\_decap\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_TUNNEL\_DECAP\_ATTR;

} sai\_tunnel\_id\_t;

/\*\*

\* @brief Attribute id for next hop

\*/

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

{

/\*\* READ-ONLY \*/

/\*\* READ-WRITE \*/

/\*\* Next hop entry type [sai\_next\_hop\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_TYPE,

/\*\* Next hop entry ipv4 address [sai\_ip\_address\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_NEXT\_HOP\_ATTR\_TYPE = SAI\_NEXT\_HOP\_IP)

\* (CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_IP,

/\*\* Next hop entry router interface id [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_NEXT\_HOP\_ATTR\_ROUTER\_INTERFACE\_ID,

/\* -- \*/

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

SAI\_NEXT\_HOP\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_next\_hop\_attr\_t;

/\*\*

\* Routine Description:

\* @brief Create next hop

\*

\* Arguments:

\* @param[out] tunnel\_id - tunnel 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

\*

\* Note: IP address expected in Network Byte Order.

\*/

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

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

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* Routine Description:

\* @brief Remove next hop

\*

\* Arguments:

\* @param[in] tunnel\_id – tunnel id

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

\_In\_ sai\_object\_id\_t tunnel\_id

);

/\*\*

\* Routine Description:

\* @brief Set Next Hop attribute

\*

\* Arguments:

\* @param[in] tunnel\_id - tunnel id

\* @param[in] attr - attribute

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

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

);

/\*\*

\* Routine Description:

\* @brief Get tunnel attribute

\*

\* Arguments:

\* @param[in] tunnel \_id - tunnel id

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

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

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* @brief Next Hop methods table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_tunnel\_api\_t

{

sai\_create\_tunnel\_fn create\_tunnel;

sai\_remove\_tunnel\_fn remove\_tunnel;

sai\_set\_tunnel\_attribute\_fn set\_tunnel\_attribute;

sai\_get\_tunnel\_attribute\_fn get\_tunnel\_attribute;

} sai\_tunnel\_api\_t;

# Examples

## IPinIP - point to point tunnel

Setup overview

Two ip networks connected via point to point ipinip tunnel details in the diagram below



Zoom in to the left side for the tunnel



Two ip networks connected via point to point ipinip tunnel details in the diagram below

**device A configuration**

Two virtual\_router , underlay VRF, overlay VRF

**overlay VRF router interface configuration**

router interface rif\_a type vlan , vlan 1

router interface rif\_lb1 type loopback (new type SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK)

local network route 192.168.0.1->rif\_a

**underlay VRF router interface configuration**

router interface rif\_lb2 type loopback (new type SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK)

local ip route 2.2.2.3->rif\_lb2

**tunnel configuration :**

tunnel ipinip\_obj

SAI\_TUNNEL\_ATTR\_TYPE = SAI\_TUNNEL\_IPINIP

SAI\_TUNNEL\_OVERLAY\_INTERFACE= rif\_lb1

SAI\_TUNNEL\_UNDERLAY\_INTERFACE= rif\_lb2

SAI\_TUNNEL\_ENCAP\_ATTR=

{

SAI\_TUNNEL\_IP\_VER=ipv4

SAI\_TUNNEL\_SRC\_IP=2.2.2.3

SAI\_TUNNEL\_TTL\_MODE= SAI\_TUNNEL\_TTL\_COPY\_FROM\_INNER

SAI\_TUNNEL\_DSCP\_MODE= SAI\_TUNNEL\_DSCP\_COPY\_FROM\_INNER

}

SAI\_TUNNEL\_DECAP\_ATTR

{

SAI\_TUNNEL\_DECAP\_SIP\_CHECK=NON

}

**Next hop configuration :**

Next hop ipinip\_decap\_obj

{

SAI\_NEXT\_HOP\_ATTR\_TYPE= SAI\_NEXT\_HOP\_TUNNEL\_DECAP(new type)

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_ID= ipinip\_obj

}

Next hop ipinip\_encap\_obj

{

SAI\_NEXT\_HOP\_ATTR\_TYPE= SAI\_NEXT\_HOP\_TUNNEL\_ENCAP(new type)

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_ID= ipinip\_obj

SAI\_NEXT\_HOP\_ATTR\_IP=3.3.3.3

}

**Route configuration:**

**overlay VRF , 192.167.0.0/16 ->**{ SAI\_ROUTE\_ATTR\_PACKET\_ACTION= FORWARD ,

SAI\_ROUTE\_ATTR\_NEXT\_HOP\_ID= ipinip\_encap\_obj

}

**ounderlay VRF , 2.2.2.3/32 ->**{ SAI\_ROUTE\_ATTR\_PACKET\_ACTION= FORWARD ,

SAI\_ROUTE\_ATTR\_NEXT\_HOP\_ID= ipinip\_decap\_obj

## VXlan

Setup overview

Overlay (virtual) network: Three tenants VA, VB, VC each tenants confine two l2 domain ( Vlans )

VA = VNI 2(map to vlan 2),VNI 3 (map to vlan 3)

Vlan 2 ip subnet = 192.168.0.0/16

Vlan 3 ip subnet = 192.167.0.0/16

VB = VNI 4(map to vlan 4),VNI 5 (map to vlan 5)

Vlan 4 ip subnet = 192.168.0.0/16

Vlan 5 ip subnet = 192.167.0.0/16

VC = VNI 400(map to vlan 6),VNI 500 (map to vlan 7)

Vlan 6 ip subnet = 192.168.0.0/16

Vlan 2 ip subnet = 192.166.0.0/16

(Since SAI doesn’t support defining of bridging domain we assume bridging domain -== vlan id -> tenants can’t share Vlan

Underlay (physical) network:

The virtual network is spared across three phy location connected via l3 network

Two are PC base hypervisor the gust Overlay (virtual) network is running on the VM on those hosts

The third location is bare metal hosts connect to a switch with VTEP endpoint

The network is described in the diagram below;



Zoom in to device A network



**device A configuration**

4 virtual\_router

underlay VRF,

overlay VRF for tenant VA\_VRF

overlay VRF for tenant VB\_VRF

overlay VRF for tenant VC\_VRF

**overlay VRF\_VA router interface configuration**

router interface rif\_va\_1 type vlan , vlan 2

router interface rif\_va\_2 type vlan , vlan 3

local network route 192.168.0.0/16-> rif\_va\_1

local network route 192.167.0.0/16-> rif\_va\_2

**overlay VRF\_VB router interface configuration**

router interface rif\_vb\_1 type vlan , vlan 4

router interface rif\_vb\_2 type vlan , vlan 5

local network route 192.168.0.0/16-> rif\_vb\_1

local network route 192.167.0.0/16-> rif\_vb\_2

**overlay VRF\_VC router interface configuration**

router interface rif\_va\_1 type vlan , vlan 5

router interface rif\_va\_2 type vlan , vlan 6

local network route 192.168.0.0/16-> rif\_vc\_1

local network route 192.166.0.0/16-> rif\_vc\_2

**underlay VRF router interface configuration**

router interface rif\_lb2 type loopback (new type SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK)

local ip route 4.4.4.3/32->rif\_lb1

**tunnel mapper configuration** // in order to be able to map VNI to vlan and vise versa

tunnel mapper vlan\_to\_vni\_map

SAI\_TUNNEL\_MAP\_ATTR\_TYPE= SAI\_TUNNEL\_MAP\_VLAN\_ID\_TO\_VNI

SAI\_TUNNEL\_MAP\_ATTR\_MAP\_TO\_VALUE\_LIST

{

Count=6

{key=2,value=2},{key=3,value=3},{key=4,value=4},

{key=5,value=5},{key=6,value=400},{key=7,value=500}

}

tunnel mapper vni\_to\_vlan\_map

SAI\_TUNNEL\_MAP\_ATTR\_TYPE= SAI\_TUNNEL\_MAP\_VNI\_TO\_VLAN\_ID

SAI\_TUNNEL\_MAP\_ATTR\_MAP\_TO\_VALUE\_LIST

{

Count=6

{key=2,value=2},{key=3,value=3},{key=4,value=4},

{key=5,value=5},{key=400,value=6},{key=500,value=7}

}

**tunnel configuration :**

tunnel vxlan\_obj

SAI\_TUNNEL\_ATTR\_TYPE = SAI\_TUNNEL\_VXLAN

SAI\_TUNNEL\_OVERLAY\_INTERFACE= NULL(should be the bridge port connected to the tunnel currently missing form SAI)

SAI\_TUNNEL\_UNDERLAY\_INTERFACE= rif\_lb1

SAI\_TUNNEL\_ENCAP\_ATTR=

{

SAI\_TUNNEL\_IP\_VER=ipv4

SAI\_TUNNEL\_IP\_VER

SAI\_TUNNEL\_SRC\_IP=4.4.4.3

SAI\_TUNNEL\_TTL\_MODE= SAI\_TUNNEL\_TTL\_USER\_DEFINE

SAI\_TUNNEL\_TTL\_VAL=1

SAI\_TUNNEL\_DSCP\_MODE= SAI\_TUNNEL\_DSCP\_USER\_DEFINE

SAI\_TUNNEL\_DSCP\_VAL=7

SAI\_TUNNEL\_ENCAP\_MAPPER= vlan\_to\_vni\_map

}

SAI\_TUNNEL\_DECAP\_ATTR

{

SAI\_TUNNEL\_DECAP\_MAPPER= vni\_to\_vlan\_map

}

**Next hop configuration :**

Next hop vxlan\_decap\_obj

{

SAI\_NEXT\_HOP\_ATTR\_TYPE= SAI\_NEXT\_HOP\_TUNNEL\_DECAP(new type)

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_ID= vxlan\_obj

}

Next hop \_encap\_obj\_host\_a

{

SAI\_NEXT\_HOP\_ATTR\_TYPE= SAI\_NEXT\_HOP\_TUNNEL\_ENCAP(new type)

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_ID= vxlan\_obj

SAI\_NEXT\_HOP\_ATTR\_IP=2.2.2.3

}

Next hop \_encap\_obj\_host\_b

{

SAI\_NEXT\_HOP\_ATTR\_TYPE= SAI\_NEXT\_HOP\_TUNNEL\_ENCAP(new type)

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_ID= vxlan\_obj

SAI\_NEXT\_HOP\_ATTR\_IP=3.3.3.3

}

**FBD configuration:**

MAC VM1,vlan 2 -> encap\_obj\_host\_a

MAC VM2,vlan 4 -> encap\_obj\_host\_a

MAC VM1,vlan 6 -> encap\_obj\_host\_b

MAC VM2,vlan 5-> encap\_obj\_host\_b

**Route configuration:**

**ounderlay VRF , 4.4.4.3/32 ->**{ SAI\_ROUTE\_ATTR\_PACKET\_ACTION= FORWARD ,

SAI\_ROUTE\_ATTR\_NEXT\_HOP\_ID= vxlan\_decap\_obj

}