

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **QOS Scheduler Group API’s** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **03/04/2015** |
| **SAI-Version** | **V0.9.3** |

**Contents**

[List of Changes i](#_Toc418962391)

[1 Overview 1](#_Toc418962392)

[1.1 Hierarchical scheduler 1](#_Toc418962393)

[1.2 Use Cases 1](#_Toc418962394)

[1.3 Default SAI Hierarchy 2](#_Toc418962395)

[1.4 Application Usage of Scheduler groups 5](#_Toc418962396)

[2 Specification 5](#_Toc418962397)

[2.1 Scheduler Group 5](#_Toc418962398)

[2.1.1 Changes to saiswitch.h 5](#_Toc418962399)

[2.1.2 Changes to saiport.h 5](#_Toc418962400)

[2.1.3 New file to saischedulergroup.h 6](#_Toc418962401)

[3 API Flow 8](#_Toc418962402)

[3.1 Example to create scheduler groups 8](#_Toc418962403)

[3.1.1 Step-1: Get number of level of hierarchy levels and group nodes supported for each level on port 9](#_Toc418962404)

[3.1.2 Step-2: Get the queue object Id’s 9](#_Toc418962405)

[3.1.3 Step-3: Create the scheduler object Id’s. 9](#_Toc418962406)

[3.1.4 Step 4: Apply scheduler to queue. 9](#_Toc418962407)

[3.1.5 Step 5: Create scheduler Group for level 1 10](#_Toc418962408)

[3.1.6 Step 6: Add queues to scheduler Group at level 1 11](#_Toc418962409)

[3.1.7 Step 7: Apply scheduler to groups at level 1. 11](#_Toc418962410)

[3.1.8 Step 8: Create scheduler Group for level 0 12](#_Toc418962411)

[3.1.9 Step 9: Add child groups at level 1 to parent group at level 0 12](#_Toc418962412)

[3.1.10 Step 10: Apply scheduler to groups at level 0. 13](#_Toc418962413)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.3 | Proposal for QOS scheduler Group API’s |  | 02/19/15 |

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

QoS enables you to provide better service to certain flows.

## Hierarchical scheduler

Hierarchical scheduler is support more queuing models and flexible binding of the schedulers, allowing the advanced traffic management. Any queuing with multiple input classes can participate in a **hierarchy**: the root scheduler node is at the top of the tree, and each of its input classes is fed by a separate lower-level schedulers. The usual understanding of hierarchical scheduling is that the non-leaf scheduler is “virtual”: they do not store data, but only make decisions as to which sub-scheduler nodes to serve. The only physical output interface is at the root, and all physical queues are at leaf group nodes.

Scheduler grouping is necessary for complex traffic management scenarios. Each level of the hierarchy will have the nodes to control the scheduler.

* Each of the level in hierarchy should be flexibly mapped to next level in hierarchy.
* Levels supported in Hierarchy is not fixed, it depends on the ASIC capability.
* There should be capability to get/set the hierarchy/grouping levels supports in ASIC.
* Each of the group nodes should supports the separately configurable min and max bandwidth guarantee and individually configured to support scheduling disciplines like SP, WRR, SP+WRR, WDRR, SP+WDRR(scheduler object) to achieve advanced traffic control.

#### Enhanced Transmission Selection (ETS)

Enhanced Transmission Selection (ETS) application in DCB supports allocation of bandwidth amongst traffic classes. When the offered load in a traffic class doesn't use its allocated bandwidth, Enhanced Transmission Selection will allow other traffic classes to use the available bandwidth.

*Traffic Class Groups:*

* A Priority Group/Traffic Class Group is a group of priorities are grouped together for the purpose of bandwidth allocation and scheduling etc.
* Application traffic with similar type is prioritized or classifies using 802.1p semantics. Eight (8) priorities are defined (0-7).

Logically ETS is subset of Hierarchical scheduler.

## Use Cases

Below are the typical scenario’s might applications interested to control by using the SAI. This depends the on the ASIC capability. But SAI needs to provide way to control in case ASIC supports.

QID=0

**Port**

QID=1

QID=2

QID=4

QID=5

QID=3

Scheduer

Scheduler

Level 3

Scheduler

Scheduler

Scheduler

Scheduler

Scheduler

Scheduler

Level 2

Scheduler

Scheduler

Level 1

Scheduler

**Sche**duler

Level 0

Scheduler

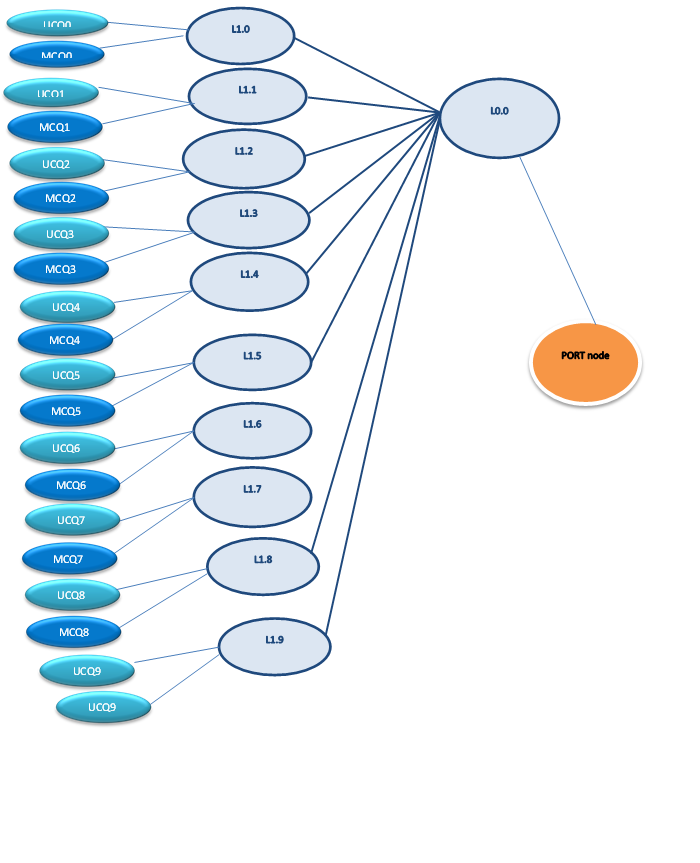
## Default SAI Hierarchy

SAI/SDK should set default hierarchy and make sure traffic is going without adaptor host (application) dependency in all the ports.

SAI Defaults:

* Number of levels N.
* Minimum 1 scheduler group per each level from Level 0 to level N-2.
* Treat unicast and multicast queue as logical pair at leaf level, in case NPU support separate unicast and multicast queue.
  + Create the scheduler groups at leaf level (Level N-1) equal to number of logical pairs of unicast and multicast queues.
* Set Level 0 parent as as port.
* Set all scheduler group parent as group at higer layer.
* This hierarchy need not be exposed to applications, this is part of documentation.

Example: Number of Levels 2 and unicast queue 9 and multicast queue 9.



## Application Usage of Scheduler groups

Application can make use of exiting default hierarchy/settings or initialize with own required hierarchy/settings.

SAI Behavior in case of applications changing the hierarchy:

* Apps will read
  + Maximum number of levels supported.
  + Maximum number of scheduler groups supported per each level.
  + Maximum number of childs per parent scheduler groups supported per level.
* Applications can start creating the scheduler groups for hierarchy.
* Whenever first scheduler group create received by the SAI at any level, SAI/SDK will remove the exiting default hierarchy and allow applications to manage the complete initiation of hierarchy in that port.
* This will make sure application have complete control of manage the all nodes in that port at all hierarchy levels.

# Specification

## Scheduler Group

### Changes to saiswitch.h

typedef struct \_sai\_scheduler\_group\_count\_t {

\_In\_ uint32\_t level;

\_Out\_ uint32\_t count;

} sai\_scheduler\_group\_count\_t;\*/

typedef enum \_sai\_switch\_attr\_t

{

..

..

/\* READ-ONLY \*/

/\*\* HQOS – Maximum Number of Hierarchy scheduler

group levels(depth) supported on all ports\*/

SAI\_SWITCH\_ATTR\_QOS\_MAX\_NUMBER\_OF\_SCHEDULER\_GROUP\_HIERARCHY\_LEVELS,

/\*\* HQOS – Maximum number of scheduler groups supported on each Hierarchy level

[sai\_scheduler\_group\_count\_t] \*/

SAI\_SWITCH\_ATTR\_QOS\_MAX\_NUMBER\_OF\_SCHEDULER\_GROUPS\_PER\_HIERARCHY\_LEVEL,

} sai\_switch\_attr\_t;

### Changes to saiport.h

typedef enum \_sai\_port\_attr\_t

{

..

/\* READ-ONLY \*/

/\* Maximum Number of child scheduler nodes can be attached to port [uint32\_t] \*/

SAI\_PORT\_ATTR\_QOS\_MAX\_SUPPORTED\_SCHEULDER\_GROUP\_CHILDS,

} sai\_port\_attr\_t;

### New file to saischedulergroup.h

/\*

\* Attribute id for Scheduler groups

\*/

typedef enum \_sai\_scheduler\_group\_attr\_t

{

/\* READ-ONLY \*/

/\* Maximum Number of childs on group [uint32\_t] \*/

SAI\_SCHEDULER\_GROUP\_ATTR\_MAX\_SUPPORTED\_CHILDS,

/\* Number of queues/groups childs added to

\* scheduler group [uint32\_t] \*/

SAI\_SCHEDULER\_GROUP\_ATTR\_CHILD\_COUNT,

/\* Scheduler Group child obejct id List [sai\_object\_list\_t] \*/

SAI\_SCHEDULER\_GROUP\_ATTR\_CHILD\_LIST,

/\* READ-WRITE \*/

/\* Scheduler group on port [sai\_object\_id\_t]

MANDATORY\_ON\_CREATE, CREATE\_ONLY \*/

SAI\_SCHEDULER\_GROUP\_ATTR\_PORT\_ID,

/\* Scheduler group level

MANDATORY\_ON\_CREATE, CREATE\_ONLY \*/

SAI\_SCHEDULER\_GROUP\_ATTR\_LEVEL,

/\* Scheucler ID [sai\_object\_id\_t] \*/

SAI\_SCHEDULER\_GROUP\_ATTR\_SCHEDULER\_PROFILE\_ID,

/\* -- \*/

/\* Custom range base value \*/

SAI\_SCHEDULER\_GROUP\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_scheduler\_group\_attr\_t;

/\*

\* Routine Description:

\* Create Scheduler group

\*

\* Arguments:

\* [out] scheduler\_group\_id – Scheudler group id

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

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

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Remove scheduler group

\*

\* Arguments:

\* [in] scheduler\_group\_id – Scheudler group id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

);

/\*

\* Routine Description:

\* Set Scheduler Group attribute

\*

\* Arguments:

\* [in] scheduler\_group\_id – scheduler group id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

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

);

/\*

\* Routine Description:

\* Get Scheduler Group attribute

\*

\* Arguments:

\* [in] scheduler\_group\_id – scheduler group id

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

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

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Add Child queue/group objects to scheduler group

\*

\* Arguments:

\* [in] parent\_id – Parent Scheduler group id

\* [in] child\_count - number of child count

\* [in] child\_objects - array of child objects

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_add\_child\_object\_to\_parent\_fn)(

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

\_In\_ uint32\_t child\_count,

\_In\_ const sai\_object\_id\_t\* child\_objects

);

/\*

\* Routine Description:

\* Remove Child queue/group objects from scheduler group

\*

\* Arguments:

\* [in] parent\_id - schedulder group id

\* [in] child\_count - number of child count

\* [in] child\_objects - array of child objects

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_child\_objects\_from\_parent\_fn)(

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

\_In\_ uint32\_t child\_count,

\_In\_ const sai\_object\_id\_t\* child\_objects

);

/\*

\* Schedulder Group methods table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_schedulder\_group\_api\_t

{

sai\_create\_schedulder\_group\_fn create\_schedulder\_group;

sai\_remove\_schedulder\_group\_fn remove\_schedulder\_group;

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

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

sai\_add\_child\_object\_to\_parent\_fn add\_child\_object\_to\_parent;

sai\_remove\_child\_objects\_from\_parent\_fn remove\_child\_object\_from\_parent;

} sai\_scheduler\_group\_api\_t;

# API Flow

## Example to create scheduler groups

Assume NPU supports

* 4 unicast queues
* 4 multicast queues
* 2 Levels of grouping supported
  + Level-1 : 2 group nodes supported
  + Level-2 : 4 group nodes are supported

### Step-1: Get number of level of hierarchy levels and group nodes supported for each level on port

uint32\_t no\_of\_level\_per\_port = 0;

sai\_object\_id\_t port\_object\_id;

sai\_attribute\_t sai\_attr\_list[5];

sai\_attr\_list[0].id = SAI\_PORT\_ATTR\_QOS\_NUMBER\_OF\_GROUP\_HIERARCHY\_LEVELS;

sai\_attr\_list[0].value = 0;

sai\_get\_switch\_attribute\_fn (1, &sai\_attr\_list);

no\_of\_level\_per\_port = sai\_attr\_list[0].value; // For this NPU, Number levels supported 2.

int no\_of\_nodes\_per\_level[2];

sai\_attribute\_t sai\_attr\_list[5];

sai\_attr\_list[0].id = SAI\_SWITCH\_ATTR\_QOS\_MAX\_NUMBER\_OF\_SCHEDULER\_GROUP\_HIERARCHY\_LEVELS;

sai\_attr\_list[0].value.level = 0;

sai\_attr\_list[0].value.count = 0;

sai\_get\_port\_attribute\_fn(port\_id, 1, &sai\_attr\_list);

no\_of\_nodes\_level[0] = sai\_attr\_list[0].value.count;

sai\_attr\_list[0].id = SAI\_SWITCH\_ATTR\_QOS\_MAX\_NUMBER\_OF\_SCHEDULER\_GROUPS\_PER\_HIERARCHY\_LEVEL;

sai\_attr\_list[0].value.level = 1;

sai\_attr\_list[0].value.count = 0;

sai\_get\_switch\_attribute\_fn(port\_id, 1, &sai\_attr\_list);

no\_of\_nodes\_level[1] = sai\_attr\_list[0].value.count;

### Step-2: Get the queue object Id’s

sai\_object\_id\_t appl\_ucast\_qid\_to\_sai\_qid[4];// Refer the queue proposal to get the sai\_qid.

sai\_object\_id\_t appl\_mcast\_qid\_to\_sai\_qid[4]; // Refer the queue proposal to get the sai\_qid.

### Step-3: Create the scheduler object Id’s.

|  |  |
| --- | --- |
| Scheduler ID | Scheduling Algorithm |
| scheduler\_id\_0 | DWRR |
| scheduler\_id\_1 | SP |

sai\_object\_id\_t scheduler\_id\_0; // Refer scheduler proposal for create scheduler object

sai\_object\_id\_t scheduler\_id\_1; // Refer scheduler proposal for create scheduler object

### Step 4: Apply scheduler to queue.

|  |  |
| --- | --- |
| Q ID | Scheduler ID |
| appl\_ucast\_qid\_to\_sai\_qid[0] | scheduler\_id\_0 |
| appl\_ucast\_qid\_to\_sai\_qid[1] | scheduler\_id\_0 |
| appl\_ucast\_qid\_to\_sai\_qid[2] | scheduler\_id\_1 |
| appl\_ucast\_qid\_to\_sai\_qid[3] | scheduler\_id\_1 |
| appl\_mcast\_qid\_to\_sai\_qid[0] | scheduler\_id\_0 |
| appl\_mcast\_qid\_to\_sai\_qid[1] | scheduler\_id\_0 |
| appl\_mcast\_qid\_to\_sai\_qid[2] | scheduler\_id\_0 |
| appl\_mcast\_qid\_to\_sai\_qid[3] | scheduler\_id\_0 |

sai\_attribute\_t sai\_attr;

sai\_attr.id = SAI\_QOS\_QUEUE\_ATTR\_SCHEDULER\_ID;

sai\_attr.value.oid = **scheduler\_id\_0**;

/\* Apply scheudler to unicast queue 0 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_ucast\_qid\_to\_sai\_qid[0], sai\_attr);

/\* Apply scheudler to unicast queue 1 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_ucast\_qid\_to\_sai\_qid[1], sai\_attr);

/\* Apply scheudler to mulcast queue 0 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_mcast\_qid\_to\_sai\_qid[0], sai\_attr);

/\* Apply scheudler to mulcast queue 1 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_mcast\_qid\_to\_sai\_qid[1], sai\_attr);

/\* Apply scheudler to mulcast queue 2 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_mcast\_qid\_to\_sai\_qid[2], sai\_attr);

/\* Apply scheudler to mulcast queue 3 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_mcast\_qid\_to\_sai\_qid[3], sai\_attr);

sai\_attr.id = SAI\_QOS\_QUEUE\_ATTR\_SCHEDULER\_ID;

sai\_attr.value.oid = **scheduler\_id\_1**;

/\* Apply scheudler to unicast queue 2 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_ucast\_qid\_to\_sai\_qid[2], sai\_attr);

/\* Apply scheudler to unicast queue 3 \*/

sai\_set\_qos\_queue\_attribute\_fn(appl\_ucast\_qid\_to\_sai\_qid[3], sai\_attr);

### Step 5: Create scheduler Group for level 1

|  |  |
| --- | --- |
| Q ID | Scheduler group |
| appl\_ucast\_qid\_to\_sai\_qid[0] | scheduler\_group\_id\_0 |
| appl\_mcast\_qid\_to\_sai\_qid[0] | scheduler\_group\_id\_0 |
| appl\_ucast\_qid\_to\_sai\_qid[1] | scheduler\_group\_id\_1 |
| appl\_mcast\_qid\_to\_sai\_qid[1] | scheduler\_group\_id\_1 |
| appl\_ucast\_qid\_to\_sai\_qid[2] | scheduler\_group\_id\_2 |
| appl\_mcast\_qid\_to\_sai\_qid[2] | scheduler\_group\_id\_2 |
| appl\_ucast\_qid\_to\_sai\_qid[3] | scheduler\_group\_id\_3 |
| appl\_mcast\_qid\_to\_sai\_qid[3] | scheduler\_group\_id\_3 |

sai\_object\_id\_t scheduler\_group\_id\_0;

sai\_object\_id\_t scheduler\_group\_id\_1;

sai\_object\_id\_t scheduler\_group\_id\_2;

sai\_object\_id\_t scheduler\_group\_id\_3;

sai\_attribute\_t sai\_attr\_list[5];

sai\_attr\_list[0].id = SAI\_SCHEDULER\_GROUP\_ATTR\_LEVEL;

sai\_attr\_list[0].value.u32 = 1;

/\* Create Group node at level 0\*/

sai\_create\_schedulder\_group\_fn (&scheduler\_group\_id\_0, 1, sai\_attr\_list);

sai\_create\_schedulder\_group\_fn (&scheduler\_group\_id\_1, 1, sai\_attr\_list);

sai\_create\_schedulder\_group\_fn (&scheduler\_group\_id\_2, 1, sai\_attr\_list);

sai\_create\_schedulder\_group\_fn (&scheduler\_group\_id\_3, 1, sai\_attr\_list);

### Step 6: Add queues to scheduler Group at level 1

/\* Add ucast 0 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_0, 1, appl\_ucast\_qid\_to\_sai\_qid[0]);

/\* Add mcast 0 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_0, 1, appl\_mcast\_qid\_to\_sai\_qid[0]);

/\* Add ucast 1 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_1, 1, appl\_ucast\_qid\_to\_sai\_qid[1]);

/\* Add mcast 1 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_1, 1, appl\_mcast\_qid\_to\_sai\_qid[1]);

/\* Add ucast 2 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_2, 1, appl\_ucast\_qid\_to\_sai\_qid[2]);

/\* Add mcast 2 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_2, 1, appl\_mcast\_qid\_to\_sai\_qid[2]);

/\* Add ucast 3 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_3, 1, appl\_ucast\_qid\_to\_sai\_qid[3]);

/\* Add mcast 3 to scheduler group \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_3, 1, appl\_mcast\_qid\_to\_sai\_qid[3]);

### Step 7: Apply scheduler to groups at level 1.

|  |  |
| --- | --- |
| Scheduler group ID | Scheduler ID |
| scheduler\_group\_id\_0 | scheduler\_id\_1 (SP) |
| scheduler\_group\_id\_1 | scheduler\_id\_0 (DWRR) |
| scheduler\_group\_id\_2 | scheduler\_id\_0 (DWRR) |
| scheduler\_group\_id\_3 | scheduler\_id\_0 (DWRR) |

sai\_attribute\_t sai\_attr;

sai\_attr.id = SAI\_SCHEDULER\_GROUP\_ATTR\_SCHEDULER\_PROFILE\_ID;

sai\_attr.value.oid = **scheduler\_id\_1**;

/\* Apply scheudler to group 0 \*/

sai\_set\_scheduler\_group\_attribute\_fn (scheduler\_group\_id\_0, sai\_attr);

sai\_attribute\_t sai\_attr;

sai\_attr.id = SAI\_SCHEDULER\_GROUP\_ATTR\_SCHEDULER\_PROFILE\_ID;

sai\_attr.value.oid = **scheduler\_id\_0**;

/\* Apply scheudler to group 1 \*/

sai\_set\_scheduler\_group\_attribute\_fn (scheduler\_group\_id\_1, sai\_attr);

/\* Apply scheudler to group 2 \*/

sai\_set\_scheduler\_group\_attribute\_fn (scheduler\_group\_id\_2, sai\_attr);

/\* Apply scheudler to group 3 \*/

sai\_set\_scheduler\_group\_attribute\_fn (scheduler\_group\_id\_3, sai\_attr);

### Step 8: Create scheduler Group for level 0

|  |  |
| --- | --- |
| Group ID | Scheduler group |
| scheduler\_group\_id\_0 | scheduler\_group\_id\_5 |
| scheduler\_group\_id\_2 |  |
| scheduler\_group\_id\_1 | scheduler\_group\_id\_6 |
| scheduler\_group\_id\_3 |  |

sai\_object\_id\_t scheduler\_group\_id\_5;

sai\_object\_id\_t scheduler\_group\_id\_6;

sai\_attribute\_t sai\_attr\_list[5];

sai\_attr\_list[0].id = SAI\_SCHEDULER\_GROUP\_ATTR\_LEVEL;

sai\_attr\_list[0].value.u32 = 2;

/\* Create Group node at level 2\*/

sai\_create\_schedulder\_group\_fn (&scheduler\_group\_id\_5, 1, sai\_attr\_list);

sai\_create\_schedulder\_group\_fn (&scheduler\_group\_id\_6, 1, sai\_attr\_list);

### Step 9: Add child groups at level 1 to parent group at level 0

/\* Add scheduler\_group\_id\_0 to scheduler group 5 \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_5, 1, scheduler\_group\_id\_0);

/\* Add scheduler\_group\_id\_2 to scheduler group 5 \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_5, 1, scheduler\_group\_id\_2);

/\* Add scheduler\_group\_id\_1 to scheduler group 5 \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_6, 1, scheduler\_group\_id\_1);

/\* Add scheduler\_group\_id\_3 to scheduler group 5 \*/

sai\_add\_child\_object\_to\_group\_fn (scheduler\_group\_id\_6, 1, scheduler\_group\_id\_3);

### Step 10: Apply scheduler to groups at level 0.

|  |  |
| --- | --- |
| Scheduler group ID | Scheduler ID |
| scheduler\_group\_id\_5 | scheduler\_id\_1 (SP) |
| scheduler\_group\_id\_6 | ssheduler\_id\_0 (DWRR) |

ai\_attribute\_t sai\_attr;

sai\_attr.id = SAI\_SCHEDULER\_GROUP\_ATTR\_SCHEDULER\_PROFILE\_ID;

sai\_attr.value.oid = **scheduler\_id\_1**;

/\* Apply scheudler to group 5 \*/

sai\_set\_scheduler\_group\_attribute\_fn (scheduler\_group\_id\_5, sai\_attr);

sai\_attribute\_t sai\_attr;

sai\_attr.id = SAI\_SCHEDULER\_GROUP\_ATTR\_SCHEDULER\_PROFILE\_ID;

sai\_attr.value.oid = **scheduler\_id\_0**;

/\* Apply scheudler to group 6 \*/

sai\_set\_scheduler\_group\_attribute\_fn (scheduler\_group\_id\_6, sai\_attr);