

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

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

**Contents**

[List of Changes i](#_Toc417392644)

[1 Overview 1](#_Toc417392645)

[1.1 Scheduler 1](#_Toc417392646)

[1.1.1 Scheduler Object 1](#_Toc417392647)

[2 Specification 3](#_Toc417392648)

[2.1 Scheduler 3](#_Toc417392649)

[2.1.1 Changes to sai.h 3](#_Toc417392650)

[2.1.2 Changes to saitypes.h 3](#_Toc417392651)

[2.1.3 New file to saischeduler.h 3](#_Toc417392652)

[2.1.4 Changes to saiqos.h 5](#_Toc417392653)

[2.1.5 Changes to saiport.h 5](#_Toc417392654)

[2.2 Configuration example 5](#_Toc417392655)

[2.2.1 Example to create schedulers for queues (Flat Hierarchy) 5](#_Toc417392656)

[2.2.2 Example for Modify schedulers with shaping params for each queue 7](#_Toc417392657)

[2.3 Example for per port shape 8](#_Toc417392658)

[2.4 Example to create schedulers for groups which are created by scheduler group object 9](#_Toc417392659)

[2.4.1 Example to create/apply schedulers for groups 9](#_Toc417392660)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.3 | Proposal for QOS scheduler 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.

## Scheduler

Schedulers arrange and/or rearrange packets for output. Scheduling is the mechanism by which packets are arranged (or rearranged) between input and output of a particular queue.

### Scheduler Object

SAI defines the Scheduler as profile to control scheduling related attributes.

Scheduler Attributes:

* Scheduling disciplines: Strict priority, DWRR.
* Weight
* Minimum bandwidth rate
* Maximum bandwidth rate

#### Strict Priority

Strict priority scheduler is to provide the min latency service to the higher Classes of traffic. Strict priority is supported for more than one queue/scheduler group.

Order of Priority between SP queues/groups:

* Queue with higher index will be schedule with high priority.
* Scheduler group priority is order of adding child to parent node. Child creation of groups. Created at last will be severed with higher priority.

#### Deficit Weighted round robin

DWRR is weighted round robin scheduler taking into account that relative weight for each queue and variable packet size state information.

#### Traffic shaping

Traffic shaping uses metering mechanisms to control the minimum and maximum bandwidth requirements for each queue/scheduler groups.

Traffic shaping aspects are tightly coupled with the scheduler. This metering mechanism that monitors the traffic flow on per queue or per scheduler group basis provides the state information whether coarse-grained flow is above or below the min and max bandwidth specifications. This state information can influence the scheduler behavior.

Based on shaping requirements, egress queues/groups are categorized as 3 scheduler groups.

MinNotMet group – This group has Queue/groups have not met their Minimum bandwidth specification.

If this group is some queues to send packets out then this group will be serviced first in their configured scheduling discipline. In case queue/group configured as SP then it will be served as SP. In case if this is configured as DWRR then it will be served as DWRR.

MaxNotMet group – This group has Queues/groups with minimum bandwidth specification satisfied and maximum bandwidth specification has not met. This group will be scheduled in case of MinNotMet group is empty and MaxNotMet group to send packets out then this group will be serviced in their configured scheduling disciplines.

MaxExceeded – Exceeded their maximum bandwidth specifications and then this group queues are not serviced.

**Example:**

Q QID=0 Buffer ID, WRED ID, SchedulerID=0

Q QID=1 Buffer ID, WRED ID SchedulerID=1

Q Node QID=1

Type = MCAST

Buffer, WRED

Q QID=2 Buffer ID, WRED ID, Scheduler ID = 1

**Port**

Q QID=4 Buffer ID, WRED ID, Scheduler ID = 2

Scheduler ID: 0

DWRR, Weight=1

Scheduler ID: 1

SP

Scheduler ID: 1

SP

Scheduler ID: 1

SP

Scheduler ID: 2

DWRR, Weight= 50, MAX BW =5Gbps

Q QID=2 Buffer ID, WRED ID, Scheduler ID=1

###### Multi-Level Hierarchy

Above Example shows the 2 level hierarchies.

#### Example for 2TCG with Hierarchy:

Q QID=3 Buffer ID, WRED ID, Scheduler ID

# Specification

## Scheduler

### Changes to sai.h

typedef enum \_sai\_api\_t

{

…

SAI\_API\_QOS\_SCHEUDLDR = 14,

} sai\_api\_t;

### Changes to saitypes.h

### New file to saischeduler.h

typedef enum \_sai\_scheduling\_type\_t {

/\* Strict Scheduling \*/

SAI\_SCHEDULING\_STRICT,

/\* Deficit Weighted Round-Robin Scheduling \*/

SAI\_SCHEDULING\_DWRR

} sai\_scheduling\_type\_t;

typedef enum \_sai\_scheduler\_attr\_t

{

/\* READ-ONLY \*/

/\* READ-WRITE \*/

/\*\* Scheduling algorithm [sai\_scheduling\_type\_t ], Default DWRR\*/

SAI\_SCHEDULER\_ATTR\_SCHEDULING\_ALGORITHM,

/\*\* [uint8\_t] scheduling algorithm weight, Range [1 - 100]

\* Valid SAI\_SCHEDULER\_ATTR\_SCHEDULING\_ALGORITHM = SAI\_SCHEDULING\_DWRR \*/

SAI\_SCHEDULER\_ATTR\_SCHEDULING\_WEIGHT,

/\* sai\_meter\_type\_t, Default bytes/sec \*/

SAI\_SCHEDULER\_ATTR\_SHAPER\_TYPE,

/\*\* [uint64\_t] Guaranteed Bandwidth shape rate [bytes/sec or PPS]

\* 0 to disable the values \*/

SAI\_SCHEDULER\_ATTR\_MIN\_BANDWIDTH\_RATE,

/\*\* [uint64\_t] Guaranteed Burst for Bandwidth shape rate [Bytes or Packets]

\*/

SAI\_SCHEDULER\_ATTR\_MIN\_BANDWIDTH\_BURST\_RATE,

/\*\* [uint64\_t] Maximum Bandwidth shape rate [bytes/sec or PPS]

\* 0 to diable the values \*/

SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_RATE,

/\*\* [uint64\_t] Maximum Burst for Bandwidth shape rate [bytes or Packets]

\*/

SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_BURST\_RATE,

/\* -- \*/

/\* Custom range base value \*/

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

} sai\_scheduler\_attr\_t;

\*

\* Routine Description:

\* Create scheduler

\*

\* Arguments:

\* [Out] sai\_scheduler\_id\_t - Scheduler 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\_fn)(

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

\_In\_ int attr\_count,

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

);

/\*

\* Routine Description:

\* Remove Scheduler

\*

\* Arguments:

\* [in] scheduler\_id – Scheduler id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

);

/\*

\* Routine Description:

\* Set Schedule node attribute

\*

\* Arguments:

\* [in] - Scheduler id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

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

);

\* Routine Description:

\* Get scheduler attribute

\*

\* Arguments:

\* [in] - Scheduler 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\_attribute\_fn)(

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

\_In\_ int attr\_count,

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

);

/\* Scheduler methods table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_scheduler\_api\_t

{

/\*\* QOS Scheduler attributes \*/

sai\_create\_scheduler\_fn create\_scheduler\_profile;

sai\_remove\_scheduler\_fn remove\_scheduler\_profile;

sai\_set\_scheduler\_attribute\_fn set\_scheduler\_attribute;

sai\_get\_scheduler\_attribute\_fn get\_scheduler\_attribute;

} sai\_scheduler\_api\_t;

### Changes to saiqos.h

typedef enum \_sai\_qos\_queue\_attr\_t

{

/\* Scheduler for queue [sai\_object\_id\_t]\*/

SAI\_QOS\_QUEUE\_ATTR\_SCHEDULER\_ID,

} sai\_qos\_queue\_attr\_t;

### Changes to saiport.h

typedef enum \_sai\_port\_attr\_t

{

/\* Scheduler for port [sai\_object\_id\_t]\*/

SAI\_PORT\_ATTR\_QOS\_SCHEDULER\_ID,

} sai\_port\_attr\_t;

## Configuration example

### Example to create schedulers for queues (Flat Hierarchy)

Below are the scheduler configurations required for queue on port 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q ID | Scheduling Algorithm | Weight | Meter Type | Min BW rate | Max BW Rate |
| 0 | DWRR | 10 |  |  |  |
| 1 | DWRR | 10 |  |  |  |
| 2 | DWRR | 80 |  |  |  |
| 3 | SP |  | Bytes |  | 5Gbps |

#### Step 1 - create schedulers with attributes values mentioned in table.

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

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

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

sai\_object\_id\_t port\_id;

sai\_object\_id\_t queue\_id\_0;

sai\_object\_id\_t queue\_id\_1;

sai\_object\_id\_t queue\_id\_2;

sai\_object\_id\_t queue\_id\_3;

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

sai\_attr\_list[0].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_ALGORITHM;

sai\_attr\_list[0].value.s32 = SAI\_SCHEDULING\_DWRR;

sai\_attr\_list[1].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_WEIGHT;

sai\_attr\_list[1].value.u8 = 10;

sai\_create\_scheduler\_fn (&scheduler\_id\_0, 2, sai\_attr\_list);

sai\_attr\_list[0].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_ALGORITHM;

sai\_attr\_list[0].value.s32 = SAI\_SCHEDULING\_DWRR;

sai\_attr\_list[1].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_WEIGHT;

sai\_attr\_list[1].value.u8 = 80;

sai\_create\_scheduler\_fn (&scheduler\_id\_2, 2, sai\_attr\_list);

sai\_attr\_list[0].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_ALGORITHM;

sai\_attr\_list[0].value.s32 = SAI\_SCHEDULING\_STRICT;

sai\_attr\_list[1].id = SAI\_SCHEDULER\_ATTR\_METER\_TYPE;

sai\_attr\_list[1].value.s32 = SAI\_METER\_TYPE\_BYTES; // Defined in saipolicer.h spec.

sai\_attr\_list[2].id = SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_RATE;

sai\_attr\_list[2].value.u64 = 5 Gbps;

sai\_create\_scheduler\_fn (&scheduler\_id\_3, 3, sai\_attr\_list);

#### Step 2 - Apply schedulers to queue 0-3 by above table

sai\_attribute\_t sai\_attr;

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

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

/\* Apply scheudler to queue 0 \*/

sai\_set\_qos\_queue\_attribute\_fn(queue\_id\_0, sai\_attr);

/\* Apply scheudler to queue 1 \*/

sai\_set\_qos\_queue\_attribute\_fn(queue\_id\_1, sai\_attr);

/\* Apply scheudler to queue 2 \*/

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

sai\_attr.value.oid = **scheduler\_id\_2**;

sai\_set\_qos\_queue\_attribute\_fn(queue\_id\_2, sai\_attr);

/\* Apply scheudler to queue 3 \*/

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

sai\_attr.value = **scheduler\_id\_3**;

sai\_set\_qos\_queue\_attribute\_fn(queue\_id\_3, sai\_attr);

### Example for Modify schedulers with shaping params for each queue

Modify existing configurations with the below shape params.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q ID | Scheduling Algorithm | Weight | Meter Type | Min BW rate | Max BW Rate |
| 0 | DWRR | 10 | Bytes | 1 Gbps | 1 Gbps |
| 1 | DWRR | 10 | Bytes | 1 Gbps | 2 Gbps |
| 2 | DWRR | 80 | Bytes | 1 Gbps | 2 Gbps |
| 3 | SP |  | Bytes |  |  |

#### Step 1- Modify scheduler with shaping attributes values mentioned in table.

/\* Set Meter type attribute to scheduler\_id\_0 \*/

sai\_attr.id = SAI\_SCHEDULER\_ATTR\_METER\_TYPE;

sai\_attr.value.s32 = SAI\_METER\_TYPE\_BYTES; // Defined in saipolicer.h spec.

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

/\* Set Min shape rate type attribute to scheduler\_id\_0 \*/

sai\_attr.id = SAI\_SCHEDULER\_ATTR\_MIN\_BANDWIDTH\_RATE;

sai\_attr.value.u64 = 1 Gbps;

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

/\* Set Max shape rate type attribute to scheduler\_id\_0 \*/

sai\_attr.id = SAI\_SCHEDULER\_ATTR\_MIN\_BANDWIDTH\_RATE;

sai\_attr.value.u64 = 1 Gbps;

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

/\* Set Meter type attribute to scheduler\_id\_2 \*/

sai\_attr.id = SAI\_SCHEDULER\_ATTR\_METER\_TYPE;

sai\_attr.value.s32 = SAI\_METER\_TYPE\_BYTES; // Defined in saipolicer.h spec.

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

/\* Set Min shape rate type attribute to scheduler\_id\_2 \*/

sai\_attr.id = SAI\_SCHEDULER\_ATTR\_MIN\_BANDWIDTH\_RATE;

sai\_attr.value.u64 = 1 Gbps;

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

/\* Set Max shape rate type attribute to scheduler\_id\_2 \*/

sai\_attr.id = SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_RATE;

sai\_attr.value.u64 = 2 Gbps;

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

/\* Remove the max rate form the queue 3, scheduler\_id\_3 \*/

/\* Set rate to 0 to remove existing rate configuration\*/

sai\_attr.id = SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_RATE;

sai\_attr.value.u64 = 0;

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

#### Step 2- Create New scheduler for queue 1 with shaping attributes values mentioned in table.

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

/\* Create new schduler for queue 1 \*/

sai\_attr\_list[0].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_ALGORITHM;

sai\_attr\_list[0].value.s32 = SAI\_SCHEDULING\_DWRR;

sai\_attr\_list[1].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_WEIGHT;

sai\_attr\_list[1].value.u8 = 10;

sai\_attr\_list[2].id = SAI\_SCHEDULER\_ATTR\_METER\_TYPE;

sai\_attr\_list[2].value.s32 = SAI\_METER\_TYPE\_BYTES; // Defined in saipolicer.h spec.

sai\_attr\_list[3].id = SAI\_SCHEDULER\_ATTR\_MIN\_BANDWIDTH\_RATE;

sai\_attr\_list[3].value.u64 = 1 Gbps;

sai\_attr\_list[4].id = SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_RATE;

sai\_attr\_list[4].value.u64 = 2 Gbps;

sai\_create\_scheduler\_fn (&scheduler\_id\_1, 4, sai\_attr\_list);

#### Step 2- Apply new scheduler to queue 1

sai\_attribute\_t sai\_attr ;

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

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

/\* Apply scheudler to queue 1 \*/

sai\_set\_qos\_queue\_attribute\_fn(queue\_id\_1, sai\_attr);

## Example for per port shape

Set the shaping on port 1 and port 2 as below.

|  |  |  |  |
| --- | --- | --- | --- |
| Port ID | Meter Type | Min BW rate | Max BW Rate |
| 0 | Bytes |  | 5 Gbps |
| 1 | Packets |  | 1 Gbps |

#### Step 1: Create New scheduler with shaping attributes values mentioned in table.

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

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

/\* Create new schduler \*/

sai\_attr\_list[0].id = SAI\_SCHEDULER\_ATTR\_METER\_TYPE;

sai\_attr\_list[0].value.s32 = SAI\_METER\_TYPE\_BYTES; // Defined in saipolicer.h spec.

sai\_attr\_list[1].id = SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_RATE;

sai\_attr\_list[1].value.u64 = 5 Gbps;

sai\_create\_scheduler\_fn (&scheduler\_id\_port\_0, 2, sai\_attr\_list);

/\* Create new schduler \*/

sai\_attr\_list[0].id = SAI\_SCHEDULER\_ATTR\_METER\_TYPE;

sai\_attr\_list[0].value.s32 = SAI\_METER\_TYPE\_PACKETS; // Defined in saipolicer.h spec.

sai\_attr\_list[1].id = SAI\_SCHEDULER\_ATTR\_MAX\_BANDWIDTH\_RATE;

sai\_attr\_list[1].value.u64 = 1 Gbps;

sai\_create\_scheduler\_fn (&scheduler\_id\_port\_1, 4, sai\_attr\_list);

#### Step 2: Apply shaping on port 0 and 1

sai\_attribute\_t sai\_attr ;

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

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

/\* Apply scheudler to port 0 \*/

sai\_set\_port\_attribute\_fn(port\_id, sai\_attr);

sai\_attribute\_t sai\_attr;

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

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

/\* Apply scheudler to port 1 \*/

sai\_set\_port\_attribute\_fn(port\_id, sai\_attr);

## Example to create schedulers for groups which are created by scheduler group object

### Example to create/apply schedulers for groups

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Group ID | Scheduling Algorithm | Weight | Meter Type | Min BW rate | Max BW Rate |
| 0 | DWRR | 50 |  |  |  |
| 1 | DWRR | 50 |  |  |  |

#### Step 1: create schedulers with attributes values mentioned in table.

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

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

sai\_attr\_list[0].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_ALGORITHM;

sai\_attr\_list[0].value.s32 = SAI\_SCHEDULING\_DWRR;

sai\_attr\_list[1].id = SAI\_SCHEDULER\_ATTR\_SCHEDULING\_WEIGHT;

sai\_attr\_list[1].value.u8 = 50;

sai\_create\_scheduler\_fn (&scheduler\_id\_g\_0, 2, sai\_attr\_list);

#### Step 2: Apply schedulers to scheduler group 0 & 1 above table

sai\_attribute\_t sai\_attr ;

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

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

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

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

/\* Apply scheduler to group 0 \*/ // Created by scheduler group object

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

/\* Apply scheudler to group 1 \*/ // Created by scheduler group object

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