

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **COS WRED API’s** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **02/19/2015** |
| **SAI-Version** | **V0.9.3** |

**Contents**

[List of Changes i](#_Toc414391295)

[1 Overview 1](#_Toc414391296)

[2 Specification 1](#_Toc414391297)

[2.1 New file saiwred.h 1](#_Toc414391298)

[2.1.1 New attributes 1](#_Toc414391299)

[2.1.2 New API’s 2](#_Toc414391300)

[2.1.3 New Method table 3](#_Toc414391301)

[2.2 Changes to saiqos.h 4](#_Toc414391302)

[2.2.1 New attributes 4](#_Toc414391303)

[2.3 Changes to saiport.h 5](#_Toc414391304)

[2.3.1 New attributes 5](#_Toc414391305)

[3 API Flow 5](#_Toc414391306)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for COS WRED – Base Version |  | 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

**Weighted random early detection** (WRED) is a queuing discipline for a network scheduler suited for congestion avoidance.It is an extension to random early detection (RED) where a single queue may have several different queue thresholds. Based on thresholds configured set for drop precedence on queue packets will be dropped or queued.

WRED proceeds in this order when a packet arrives:

* Calculation of the average queue size.
* The arriving packet is queued only if the average queue size is below the minimum queue threshold.
* Depending on the packet drop probability the packet is either dropped or queued if the average queue size is between the minimum and maximum queue threshold.
* The packet is automatically dropped if the average queue size is greater than the maximum threshold.

**Calculation of average queue size:**

The average queue size depends on the previous average as well as the current size of the queue. The calculation formula is given below:

 avg= o*(1-2^{-n}) + c*(2^{-n})\,\!

Where n is the user-configurable exponential weight factor, o is the old average and c is the current queue length. The previous average will be more important for high values of n. Peaks and Lows in queue length will be smoothed by a high value. Low values of n allow the average queue size to stay close to the current queue size.

This object defines the various attributes needs to configure WRED.

**This object can be used to enable WRED on per queue, per queue group, per port and port buffer pools.**

# Specification

## New file saiwred.h

### New attributes

/\*\* WRED \*/

typedef enum \_sai\_wred\_attr\_t

{

/\* [bool] enable/disable, Default FALSE\*/

SAI\_WRED\_ATTR\_GREEN\_ENABLE,

/\*\* Bytes [ sai\_uint32\_t], MANDATORY for SAI\_WRED\_ATTR\_GREEN\_ENABLE = TRUE \*/

SAI\_WRED\_ATTR\_GREEN\_MIN\_THRESHOLD,

/\*\* bytes [sai\_uint32\_t], MANDATORY for SAI\_WRED\_ATTR\_GREEN\_ENABLE = TRUE \*/

SAI\_WRED\_ATTR\_GREEN\_MAX\_THRESHOLD,

/\*\* Percentage 0 – 100 [sai\_uint32\_t], Default 100% \*/

SAI\_WRED\_ATTR\_GREEN\_DROP\_PROBABILITY,

/\* [bool] enable/disable per color, Default FALSE \*/

SAI\_WRED\_ATTR\_YELLOW\_ENABLE,

/\*\* bytes [ sai\_uint32\_t], MANDATORY for SAI\_WRED\_ATTR\_YELLOW\_ENABLE = TRUE \*/

SAI\_WRED\_ATTR\_YELLOW\_MIN\_THRESHOLD,

/\*\* bytes [sai\_uint32\_t], MANDATORY for SAI\_WRED\_ATTR\_YELLOW\_ENABLE = TRUE \*/

SAI\_WRED\_ATTR\_YELLOW\_MAX\_THRESHOLD,

/\*\* Percentage 0 – 100 [sai\_uint32\_t], Default 100%\*/

SAI\_WRED\_ATTR\_YELLOW\_DROP\_PROBABILITY,

/\* [bool] enable/disable per RED color, Default FALSE \*/

SAI\_WRED\_ATTR\_RED\_ENABLE,

/\*\* bytes [ sai\_uint32\_t], MANDATORY for SAI\_WRED\_ATTR\_RED\_ENABLE = TRUE \*/

SAI\_WRED\_ATTR\_RED\_MIN\_THRESHOLD,

/\*\* bytes [ sai\_uint32\_t], MANDATORY for SAI\_WRED\_ATTR\_RED\_ENABLE = TRUE \*/

SAI\_WRED\_ATTR\_RED\_MAX\_THRESHOLD,

/\*\* Percentage 0 – 100 [sai\_uint32\_t], Default 100%\*/

SAI\_WRED\_ATTR\_RED\_DROP\_PROBABILITY,

/\*\* 0 – 15 [sai\_uint8\_t], Default 0\*/

SAI\_WRED\_ATTR\_WEIGHT,

/\* [bool] enable/disable ECN marking, Default is FALSE \*/

SAI\_WRED\_ATTR\_ECN\_MARK\_ENABLE,

} sai\_wred\_attr\_t;

### New API’s

/\*

\* Routine Description:

\* Create WRED Object

\*

\* Arguments:

\* [Out] wred\_id - WRED 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\_wred\_fn)(

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

\_In\_ int attr\_count,

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

);

/\*

\* Routine Description:

\* Remove WRED Object

\*

\* Arguments:

\* [in] wred\_id - WRED id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

);

/\*

\* Routine Description:

\* Set WRED attributes

\*

\* Arguments:

\* [in] wred\_id – WRED id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

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

);

/\*

\* Routine Description:

\* Get WRED attribute

\*

\* Arguments:

\* [in] wred\_id - WRED profile 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\_wred\_attribute\_fn)(

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

\_In\_ int attr\_count,

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

);

### New Method table

/\*

\* WRED methods table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_wred\_api\_t

{

sai\_create\_wred\_fn create\_wred\_profile;

sai\_remove\_wred\_fn remove\_wred\_profile;

sai\_set\_wred\_attribute\_fn set\_wred\_profile\_attribute;

sai\_get\_wred\_attribute\_fn get\_wred\_profile\_attribute;

} sai\_wred\_api\_t;

## Changes to saiqos.h

### New attributes

/\* Drop type \*/

typedef enum \_sai\_qos\_drop\_type\_t {

/\* TAIL DROP \*/

SAI\_QOS\_DROP\_TYPE\_TAIL,

/\* weighted Random early drop\*/

SAI\_QOS\_DROP\_TYPE\_WRED,

} sai\_qos\_drop\_type\_t;

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

{

/\* Drop Type [sai\_qos\_drop\_type\_t],

Default (TAIL DROP) \*/

SAI\_QOS\_QUEUE\_ATTR\_DROP\_TYPE,

/\* Attach WRED profile ID to queue [sai\_object\_id\_t]

Mandatory when SAI\_QOS\_QUEUE\_ATTR\_DROP\_TYPE = SAI\_QOS\_DROP\_TYPE\_WRED \*/

SAI\_QOS\_QUEUE\_ATTR\_WRED\_ID,

} sai\_qos\_queue\_attr\_t ;

typedef enum \_sai\_qos\_queue\_stat\_counter\_t

{

/\* get/set WRED green color dropped packets count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_GREEN\_DISCARD\_DROPPED\_PACKETS,

/\* get/set WRED green color dropped bytes count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_GREEN\_DISCARD\_DROPPED\_BYTES,

/\* get/set WRED yellow color dropped packets count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_YELLOW\_DISCARD\_DROPPED\_PACKETS,

/\* get/set WRED yellow color dropped bytes count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_YELLOW\_DISCARD\_DROPPED\_BYTES,

/\* get/set WRED red color dropped packets count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_RED\_DISCARD\_DROPPED\_PACKETS,

/\* get/set WRED red color dropped bytes count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_RED\_DISCARD\_DROPPED\_BYTES,

/\* get/set WRED dropped packets count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_DISCARD\_DROPPED\_PACKETS,

/\* get/set WRED red dropped bytes count [uint64\_t] \*/

SAI\_QOS\_QUEUE\_STAT\_DISCARD\_DROPPED\_BYTES

} sai\_qos\_queue\_stat\_counter\_t;

typedef enum \_sai\_port\_stat\_counter\_t

{

/\* get/set WRED green packet count [uint64\_t] \*/

SAI\_PORT\_STAT\_GREEN\_DISCARD\_DROPPED\_PACKETS,

/\* get/set green byte count [uint64\_t] \*/

SAI\_PORT\_STAT\_GREEN\_DISCARD\_DROPPED\_BYTES,

/\* get/set WRED yellow packet count [uint64\_t] \*/

SAI\_PORT\_STAT\_YELLOW\_DISCARD\_DROPPED\_PACKETS,

/\* get/set WRED yellow byte count [uint64\_t] \*/

SAI\_PORT\_STAT\_YELLOW\_DISCARD\_DROPPED\_BYTES,

/\* get/set WRED red packet count [uint64\_t] \*/

SAI\_PORT\_STAT\_RED\_DISCARD\_DROPPED\_PACKETS,

/\* get/set WRED red byte count [uint64\_t] \*/

SAI\_PORT\_STAT\_RED\_DISCARD\_DROPPED\_BYTES,

SAI\_PORT\_STAT\_DISCARD\_DROPPED\_PACKETS,

SAI\_PORT\_STAT\_DISCARD\_DROPPED\_BYTES

} sai\_port\_stat\_counter\_t;

## Changes to saiport.h

### New attributes

typedef enum \_sai\_port\_attr\_t

{

/\* sai\_qos\_drop\_type\_t, Default (TAIL DROP)\*/

SAI\_PORT\_ATTR\_QOS\_DROP\_TYPE,

/\* Attach WRED to port [sai\_object\_id\_t]

(mandatory when SAI\_PORT\_ATTR\_QOS\_DROP\_TYPE = SAI\_QOS\_DROP\_TYPE\_WRED) \*/

SAI\_PORT\_ATTR\_QOS\_WRED\_ID

} sai\_port\_attr\_t;

## Configuration examples

### WRED object create

sai\_object\_id\_t wred\_id;

sai\_attribute\_t attr\_list[4];

int attr\_count = 4;

attr\_list[0].id = SAI\_WRED\_ATTR\_GREEN\_ENABLE;

attr\_list[0].value.booldata = true;

attr\_list[1].id = SAI\_WRED\_ATTR\_GREEN\_MIN\_THRESHOLD;

attr\_list[1].value.s32 = 50;

attr\_list[2].id = SAI\_WRED\_ATTR\_GREEN\_MAX\_THRESHOLD;

attr\_list[2].value.s32 = 100;

attr\_list[3].id = SAI\_WRED\_ATTR\_GREEN\_DROP\_PROBABILITY;

attr\_list[3].value.s32 = 30;

sai\_create\_wred\_fn (&wred\_id, attr\_count, p\_attr\_list);

### Applying the wred to a queue

sai\_object\_id\_t port\_id = 10;

sai\_object\_id\_t queue\_id = 10 ;

sai\_attribute\_t sai\_queue\_attr\_set;

sai\_queue\_attr\_set.id = SAI\_QOS\_QUEUE\_ATTR\_DROP\_TYPE ;

sai\_queue\_attr\_set.value.s32 = SAI\_QOS\_DROP\_TYPE\_WRED ;

sai\_set\_queue\_attribute\_fn (port\_id, queue\_id, &sai\_queue\_attr\_set);

sai\_queue\_attr\_set.id = SAI\_QOS\_QUEUE\_ATTR\_WRED\_ID;

sai\_queue\_attr\_set.value.oid = wred\_id;

sai\_set\_queue\_attribute\_fn (port\_id, queue\_id, &sai\_queue\_attr\_set);

### Applying the wred to a port

sai\_object\_id\_t port\_id = 0;

sai\_attribute\_t sai\_port\_attr\_set;

sai\_port\_attr\_set.id = SAI\_PORT\_ATTR\_QOS\_DROP\_TYPE;

sai\_port\_attr\_set.value.s32 = SAI\_QOS\_DROP\_TYPE\_WRED ;

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

sai\_port\_attr\_set.id = SAI\_PORT\_ATTR\_QOS\_WRED\_ID;

sai\_port\_attr\_set.value.oid = wred\_id;

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