

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **SAI TAM Enhancements for Monitoring Microbursts** |
| **Authors** | **Marvell Semiconductor Ltd.** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **25 Feb 2017** |
| **SAI-Version** | **1.2** |

Contents

[List of Changes 3](#_Toc476856278)

[1 Overview 5](#_Toc476856279)

[2 Proposal 5](#_Toc476856280)

[2.1 TAM Microburst Objects 5](#_Toc476856281)

[2.2 Microburst statistics 6](#_Toc476856282)

[3 Specification 6](#_Toc476856283)

[3.1 Addition to file saitam.h 6](#_Toc476856284)

[3.1.1 Data Structures and Enumerations 6](#_Toc476856285)

[3.1.2 API 10](#_Toc476856286)

[3.2 Changes to saitypes.h 11](#_Toc476856287)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 1 | Proposal for SAI TAM Enhancements for Monitoring Microbursts | Marvell Semiconductor Ltd. | 25 Feb 2017 |
| 2 | Update after SAI workshop |  |  |
| 3 | Adding histogram object, examples |  | 12 May 2017 |

License

© 2017 Microsoft Corporation, Dell Inc., Facebook, Inc, Broadcom Limited, Intel Corporation, Marvell Semiconductor Ltd., 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

SAI has “Telemetry and Analytics” (TAM) provision for retrieving statistics data in bulk, as well as introduces a notification mechanism for monitoring statistics using thresholds.

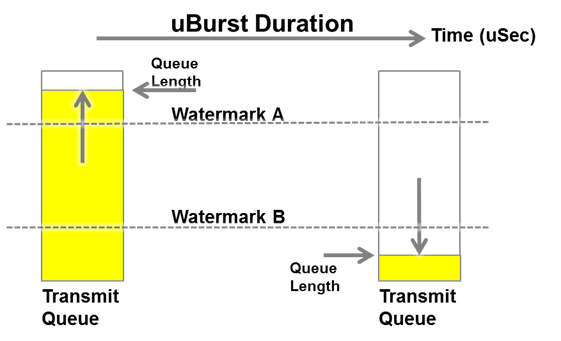
This document proposes a TAM enhancements for monitoring microbursts - retrieving statistics data on microbursts and microbursts durations. It uses the retrieving mechanism introduced in TAM Specification.

# Proposal

## TAM Microburst Objects

This document enhances TAM by adding Microburst objects, way to manage them and retrieving associated statistics.

The Microburst (uBurst) is an event in which a buffer-count (e.g., a queue length) crosses watermark A (from low to high) until it crosses watermark B (from high to low).



**Figure 1 Microburst Definition**

This spec supports microburst configuration via Microburst objects. Microburst objects are created via the **sai\_tam\_microburst\_create\_fn** API. The statistic associated with the threshold object and the associated breach levels are passed as attributes during the microburst object creation.

## Microburst statistics

Microburst Duration is the time (in us) of microburst event (from cross A until it crosses B).

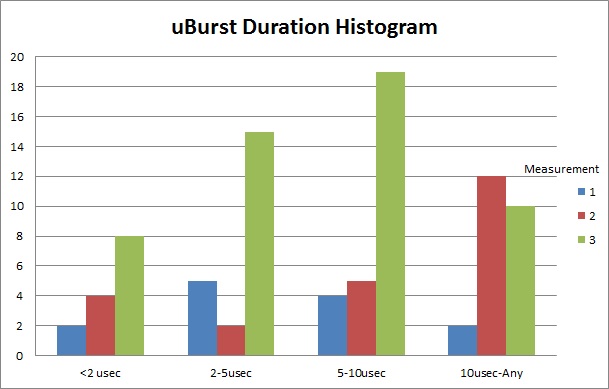
Microburst Duration Statistics:

* Last microburst duration
* Longest microburst duration (peak)
* Shortest microburst duration (min)
* Average duration of all microbursts
* Number of microbursts
* Microburst Durations histogram

Microburst Durations histogram statistics contain numbers of microbursts according to their durations in user-defined intervals - see Figure 2 with example of a histogram with four bins:

* Duration-bin-#0 - number of microbursts with duration from t0=0 to t1=2us
* Duration-bin-#1 - number of microbursts with duration from t1=2us to t2=5us
* Duration-bin-#2 - number of microbursts with duration from t2=5us to t3=10us
* Duration-bin-#3 - number of microbursts with any duration more than t3=10us.

Three colours of the bars represent three measurements made by application, for example every one second.



**Figure 2 Example of Microburst Duration Histogram**

# Specification

## Addition to file saitam.h

### Data Structures and Enumerations

/\*\*

\* @brief Enum defining statistics for microburst.

\*/

typedef enum \_sai\_tam\_microburst\_stat\_t

{

/\*\* get/set last uBurst duration in us [uint64\_t] \*/

SAI\_TAM\_MICROBURST\_STAT\_LAST\_DURATION = 0x00000000,

/\*\* get/set longest uBurst duration in us [uint64\_t] \*/

SAI\_TAM\_MICROBURST\_STAT\_LONGEST\_DURATION = 0x00000001,

/\*\* get/set shortest uBurst duration in us [uint64\_t] \*/

SAI\_TAM\_MICROBURST\_STAT\_SHORTEST\_DURATION = 0x00000002,

/\*\* get/set average uBurst duration in us [uint64\_t] \*/

SAI\_TAM\_MICROBURST\_STAT\_AVERAGE\_DURATION = 0x00000003,

/\*\* get/set number of uBursts [uint64\_t] \*/

SAI\_TAM\_MICROBURST\_STAT\_NUMBER = 0x00000004,

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

SAI\_TAM\_MICROBURST\_STAT\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_tam\_microburst\_stat\_t;

/\*\*

\* @brief TAM Microburst Attributes.

\*/

typedef enum \_sai\_tam\_microburst\_attr\_t {

/\*\*

\* @brief Start of Attributes

\*/

SAI\_TAM\_MICROBURST\_ATTR\_START,

/\*\*

\* @brief TAM Object

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE

\* @objects SAI\_OBJECT\_TYPE\_TAM

\*/

SAI\_TAM\_MICROBURST\_ATTR\_TAM\_ID = SAI\_TAM\_MICROBURST\_ATTR\_START,

/\*\*

\* @brief Statistic for this microburst

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE

\* @objects SAI\_OBJECT\_TYPE\_TAM\_STAT

\*/

SAI\_TAM\_MICROBURST\_ATTR\_STATISTIC,

/\*\*

\* @brief Watermark Levels

\*

\* Breach high and low levels for this microburst in number of bytes

\*

\* If not specified, the microburst is created without any

\* levels, which is effectively disabling the microburst

\* monitoring for the statistic

\*

\* @type uint64\_t

\* @flags CREATE\_AND\_SET

\* @default 0

\*/

SAI\_TAM\_MICROBURST\_ATTR\_LEVEL\_HIGH,

SAI\_TAM\_MICROBURST\_ATTR\_LEVEL\_LOW,

/\*\*

\* @brief Transporter Object

\*

\*

\* @type sai\_object\_id\_t

\* @allownull true

\*/

SAI\_TAM\_MICROBURST\_ATTR\_TRANSPORTER,

/\*\*

\* @brief Buffers/Statistics for inclusion in the snapshot

\* Specifies the Statistics/Types for the snapshot.

\* If not specified, all buffers tracked by

\* the associated TAM object are included in the snapshot.

\* When specified, the buffers requested for snapshot must be within the set

\* tracked by the associated TAM object.

\*

\* @type sai\_tam\_statistic\_id\_list

\* @allownull true

\*/

SAI\_TAM\_MICROBURST\_ATTR\_STAT\_TYPES,

/\*\*

\* @brief End of Attributes

\*/

SAI\_TAM\_MICROBURST\_ATTR\_END

} sai\_tam\_microburst\_attr\_t;

/\*\*

\* @brief TAM Histogram Attributes.

\*/

typedef enum \_sai\_tam\_histogram\_attr\_t

{

/\*\*

\* @brief Start of Attributes

\*/

SAI\_TAM\_HISTOGRAM\_ATTR\_START,

/\*\*

\* @brief TAM Object

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_AND\_SET

\* @objects SAI\_OBJECT\_TYPE\_TAM

\*/

SAI\_TAM\_HISTOGRAM\_ATTR\_TAM\_ID = SAI\_TAM\_HISTOGRAM\_ATTR\_START,

/\*\*

\* @brief Statistic for this histogram

\*

\* @type sai\_uint32\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_AND\_SET

\* @default 0

\*/

SAI\_TAM\_HISTOGRAM\_ATTR\_NUMBER\_OF\_BINS,

/\*\*

\* @brief Histogram Bins Lower Boundaries

\*

\* List of lower boundary of each bin for this HISTOGRAM in

\* number refered object units. The upper boundary of a bin is

\* the lower boundary of next bin. The upper boundary of the

\* last bin is infinity.

\*

\* @type sai\_object\_list\_t of sai\_uint32\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_AND\_SET

\* @default empty

\*/

SAI\_TAM\_HISTOGRAM\_ATTR\_BIN\_BOUNDARY,

/\*\*

\* @brief Transporter Object

\*

\* Provides the transporter object for this histogram. When the

\* data shot is made, this transporter will be used to 'copy'

\* the data to the 'transporter-desired' location. In the

\* absence of a transporter, the tracker's default transporter

\* will be used (DEFAULT).

\*

\* @type sai\_object\_id\_t

\* @flags CREATE\_AND\_SET

\* @objects SAI\_OBJECT\_TYPE\_TAM\_TRANSPORTER

\* @allownull true

\* @default SAI\_NULL\_OBJECT\_ID

\*/

SAI\_TAM\_HISTOGRAM\_ATTR\_TRANSPORTER,

/\*\*

\* @brief End of Attributes

\*/

SAI\_TAM\_HISTOGRAM\_ATTR\_END,

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

SAI\_TAM\_HISTOGRAM\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\*\* End of custom range base \*/

SAI\_TAM\_HISTOGRAM\_ATTR\_CUSTOM\_RANGE\_END

} sai\_tam\_histogram\_attr\_t;

### API

/\*\*

\* Routine Description:

\* @brief Create and return a microburst object

\*

\* This creates a microburst in the hardware with the associated statistic

\* passed via the attributes.

\*

\* Arguments:

\* @param[out] microburst\_id - microburst object

\* @param[in] attr\_list – preferences for creating a microburst

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t(\*sai\_tam\_microburst\_create\_fn) (

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

\_In\_ uint32\_t attr\_count,

\_In\_ sai\_attribute\_t \* attr\_list /\* sai\_tam\_microburst\_attr\_t \*/

);

/\*\*

\* Routine Description:

\* @brief Deletes a specified microburst object.

\*

\* Arguments:

\* @param[in] microburst\_id - microburst object to be removed.

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t(\*sai\_tam\_microburst\_remove\_fn) (

\_In\_ sai\_object\_id\_t \* microburst\_id

);

/\*\*

\* Routine Description:

\* @brief Get values for specified microburst attributes.

\*

\* Arguments:

\* @param[in] microburst\_id – microburst object 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\_tam\_microburst\_attribute\_get\_fn) (

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

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* Routine Description:

\* @brief Set microburst attribute value(s).

\*

\* Arguments:

\* @param[in] microburst\_id - microburst object id

\* @param[in] attr\_list - attribute

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t(\*sai\_tam\_microburst\_attribute\_set\_fn) (

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

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* @brief Create and return a histogram object

\*

\* This creates a histogram in the driver. Via the attributes,

\* caller may indicate a preference for histogram of a specific

\* set of bins.

\*

\* @param[out] tam\_histogram\_id histogram object

\* @param[in] switch\_id Switch object 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\_tam\_histogram\_fn)(

\_Out\_ sai\_object\_id\_t \*tam\_histogram\_id,

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

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief Deletes a specified histogram object and free driver memory.

\*

\* @param[in] tam\_histogram\_id histogram object to be removed.

\*

\* @return #SAI\_STATUS\_SUCCESS on success, failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_tam\_histogram\_fn)(

\_In\_ sai\_object\_id\_t tam\_histogram\_id);

/\*\*

\* @brief Set histogram attribute value(s).

\*

\* @param[in] tam\_histogram\_id histogram object id

\* @param[in] attr Attribute to set

\*

\* @return #SAI\_STATUS\_SUCCESS on success, failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_tam\_histogram\_attribute\_fn)(

\_In\_ sai\_object\_id\_t tam\_histogram\_id,

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

/\*\*

\* @brief Get values for specified histogram attributes.

\*

\* @param[in] tam\_histogram\_id histogram object 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\_tam\_histogram\_attribute\_fn)(

\_In\_ sai\_object\_id\_t tam\_histogram\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief Obtain the values for all bins from a histogram.

\*

\* Attribute list must supply sufficient memory for values of

\* all bins as specified for the histogram object.

\*

\* @param[in] tam\_histogram\_id histogram object id

\* @param[inout] number\_of\_bins Number of ins(required/provided)

\* @param[inout] values Statistics (allocated/provided)

\*

\* @return #SAI\_STATUS\_SUCCESS on success, failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_tam\_histogram\_stats\_fn)(

\_In\_ sai\_object\_id\_t tam\_histogram\_id,

\_Inout\_ uint32\_t \*number\_of\_bins,

\_Inout\_ uint32\_t \*bin\_values);

## Add to sai\_tam\_api\_t:

/\* Microburst object API \*/

sai\_tam\_microburst\_create\_fn tam\_microburst\_create;

sai\_tam\_microburst\_remove\_fn tam\_microburst\_remove;

sai\_tam\_microburst\_attribute\_set\_fn tam\_microburst\_attribute\_set;

sai\_tam\_microburst\_attribute\_get\_fn tam\_microburst\_attribute\_get;

sai\_create\_tam\_histogram\_fn create\_tam\_histogram;

sai\_remove\_tam\_histogram\_fn remove\_tam\_histogram;

sai\_set\_tam\_histogram\_attribute\_fn set\_tam\_histogram\_attribute;

sai\_get\_tam\_histogram\_attribute\_fn get\_tam\_histogram\_attribute;

sai\_get\_tam\_histogram\_stats\_fn get\_tam\_histogram\_stats;

## Changes to saitypes.h

The following are the changes to the saitypes.h file.

* The union **sai\_object\_type\_t** is enhanced to include the following new object types:
  + **SAI\_OBJECT\_TYPE\_TAM\_HISTOGRAM**,
  + **SAI\_OBJECT\_TYPE\_TAM\_MICROBURST**,

# Examples

## Create Microburst

sai\_object\_id\_t tam\_id;

sai\_object\_id\_t tam\_histogram\_id;

sai\_object\_id\_t microburst\_id;

sai\_object\_id\_t switch\_id;

sai\_object\_id\_t queue\_id;

sai\_object\_id\_t statistic\_id;

uint32\_t attr\_count;

sai\_attribute\_t attr\_list[16];

attr\_list[0].id = (sai\_attr\_id\_t)SAI\_TAM\_MICROBURST\_ATTR\_TAM\_ID;

attr\_list[0].value.oid = tam\_id;

attr\_list[1].id = (sai\_attr\_id\_t)SAI\_TAM\_MICROBURST\_ATTR\_STATISTIC;

attr\_list[1].value.oid = tam\_stat\_id;

attr\_list[2].id = SAI\_TAM\_MICROBURST\_ATTR\_LEVEL\_A;

attr\_list[2].value.uint64\_t = 50000;

attr\_list[3].type = SAI\_TAM\_MICROBURST\_ATTR\_LEVEL\_B;

attr\_list[3].value.uint64\_t = 450000;

attr\_list[4].type = SAI\_TAM\_MICROBURST\_ATTR\_STATISTIC;

attr\_list[4].value = statistic\_id;

attr\_list[5].type = SAI\_TAM\_HISTOGRAM\_ATTR\_TRANSPORTER;

attr\_list[5].value = 0; /\* GET/polling by Host Adapter \*/

attr\_count = 6;

tam\_microburst\_create(

&microburst\_id,

attr\_count,

attr\_list);

## Create Histogram

attr\_list[0].type = SAI\_TAM\_HISTOGRAM\_ATTR\_TAM\_ID;

attr\_list[0].value.oid = microburst\_id;

attr\_list[1].type = SAI\_TAM\_HISTOGRAM\_ATTR\_NUMBER\_OF\_BINS;

attr\_list[1].value,uint32 = 4;

attr\_list[2].type = SAI\_TAM\_HISTOGRAM\_ATTR\_BIN\_BOUNDARY;

attr\_list[2].value.list[0] = 0; /\* bin-#0 - number of microbursts with duration from 0us to 2us \*/

attr\_list[2].value.list[1] = 2; /\* bin-#1 - number of microbursts with duration from t1=2us to t2=5us \*/

attr\_list[2].value.list[2] = 5; /\* in-#2 - number of microbursts with duration from t2=5us to t3=10us \*/

attr\_list[2].value.list[3] = 10; /\* bin-#3 - number of microbursts with any duration more than t3=10us. \*/

attr\_list[2].value.list\_size=4

attr\_list[3].type = SAI\_TAM\_HISTOGRAM\_ATTR\_TRANSPORTER;

attr\_list[3].value = 0; /\* GET polling by Host Adapter \*/

attr\_count = 4;

create\_tam\_histogram(

&tam\_histogram\_id,

switch\_id,

attr\_count,

attr\_list);

## Get Microburst Statistics

attr\_list[0].type = SAI\_TAM\_MICROBURST\_STAT\_LAST\_DURATION; /\* last uBurst duration in us \*/

attr\_list[1].type = SAI\_TAM\_MICROBURST\_STAT\_LONGEST\_DURATION; /\* longest uBurst duration in us from last get \*/

attr\_list[2].type = SAI\_TAM\_MICROBURST\_STAT\_SHORTEST\_DURATION; /\* shortest uBurst duration in us from last get \*/

attr\_list[3].type = SAI\_TAM\_MICROBURST\_STAT\_AVERAGE\_DURATION; /\* average uBurst duration in us from last get \*/

attr\_list[4].type = SAI\_TAM\_MICROBURST\_STAT\_NUMBER; /\* number of uBursts from last get \*/

attr\_count = 5;

microburst\_attribute\_get (

microburst\_id,

attr\_count,

attr\_list);

## Get Histogram Data

uint32\_t number\_of\_bins = 16;

uint32\_t bin\_values[16];

/\* \*/

get\_tam\_histogram\_stats(

tam\_histogram\_id,

&number\_of\_bins,

bin\_values);

## Delete Histogram and Microburst Objects

tam\_microburst\_remove(microburst\_id);

remove\_tam\_histogram(histogram\_id);