

|  |
| --- |
|  |

QoS VoQ Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **QoS VoQ** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **07/27/2016** |
| **SAI-Version** | **V0.9.3** |

**Contents**

[List of Changes i](#_Toc458102506)

[1 Overview 1](#_Toc458102507)

[1.1 Introduction 1](#_Toc458102508)

[1.1.1 Unicast VOQ 1](#_Toc458102509)

[1.1.2 Multicast VOQ 2](#_Toc458102510)

[2 Specification 3](#_Toc458102511)

[2.1 Changes to saiqueue.h 3](#_Toc458102512)

[2.2 Changes to saiqosmaps.h 4](#_Toc458102513)

[3 Configuration Example 5](#_Toc458102514)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| Initial Version | QoS extension to support VoQ (Virutal Output Queue) | Ravikumar Sivasankar | 27 July 2016 |
| 0.2 | Corrected the typo. | Ravikumar Sivasankar | 05 Aug 2016 |
| 0.3 | Corrected the typo. | Ravikumar Sivasankar | 05 Aug 2016 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

Definition and Acronyms

|  |  |
| --- | --- |
| VoQ | Virtual Output Queue |
| HOL | Head of Line |
|  |  |

## Introduction

### Unicast VOQ

Unicast Virtual Output Queues (VOQ) are Unicast Ingress Queues assigned to each output port in the system. The output port refers to the global system port in the multi-npu (multi chassis) system.

In a non VoQ system, common problem that occurs is Head-Of-Line (HOL) blocking.

In following diagram, consider that Ingress Port P1 transmits traffic to Egress Port P3 and P4.



If Egress Port P3 is congested, then it sends back pressure indication to the Ingress Port P1. As a result, the traffic from Ingress Port P1 destined to Egress Port P4 cannot be sent until the congestion at P3 clears.

Virtual Output Queues eliminates the HOL Blocking as explained below.



Consider the same scenario where the Ingress Port P1 transmits traffic to Egress Port P3 and P4.

If Egress Port P3 is congested, then it will not affect the traffic from P1 destined to Egress Port P4, since the traffic will be queued in the VOQs dedicated to the Port P4.

### Multicast VOQ

Multicast VOQ queues are Ingress Multicast queues. These queues can be either global or per destination Switch based, depending upon the NPU capabilities.

If it is destination Switch based, then each Switch will maintain a set of Multicast queues for each destination Switch.

If it is not destination Switch based, then each Switch will maintain a set of Multicast queues globally.

# Specification

The changes are highlighted in blue.

## Changes to saiqueue.h

/\*\*

\* @brief Enum defining Queue types.

\*/

typedef enum \_sai\_queue\_type\_t

{

/\*\* H/w Queue for all types of traffic \*/

SAI\_QUEUE\_TYPE\_ALL = 0x00000000,

/\*\* H/w Egress Unicast Queue \*/

SAI\_QUEUE\_TYPE\_UNICAST = 0x00000001,

/\*\* H/w Multicast Egress (Broadcast, Unknown unicast, Multicast) Queue \*/

SAI\_QUEUE\_TYPE\_MULTICAST = 0x00000002,

/\*\* H/w Virtual Output Queue (VoQ). This queue is ingress unicast queue \*/

SAI\_QUEUE\_TYPE\_UNICAST\_VOQ = 0x00000003,

/\*\* H/w Ingress Multicast Queue. \*/

SAI\_QUEUE\_TYPE\_MULTICAST\_VOQ = 0x00000004,

/\* -- \*/

/\* Custom range base value \*/

SAI\_QUEUE\_TYPE\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_queue\_type\_t;

typedef enum \_sai\_queue\_attr\_t

{

/\*\* READ-ONLY \*/

/\*\* Queue type [sai\_queue\_type\_t]

\* (CREATE\_ONLY|MANDATORY\_ON\_CREATE|KEY) \*/

SAI\_QUEUE\_ATTR\_TYPE = 0x00000000,

/\* Queue index [sai\_uint8\_t]

\* (CREATE\_ONLY|MANDATORY\_ON\_CREATE|KEY) \*/

SAI\_QUEUE\_ATTR\_INDEX,

/\* Queue attachment point [sai\_object\_id\_t].

\* (CREATE\_ONLY|MANDATORY\_ON\_CREATE|KEY)

\* The object the Queue is attached to. It should be one of

\* Physical port object id, Switch object id or Null object id.

\* If Queue Type is sai port\_object (only Physical ports) for

\* SAI\_QUEUE\_TYPE\_UNICAST, SAI\_QUEUE\_TYPE\_MULTICAST and SAI\_QUEUE\_TYPE\_UNICAST\_VOQ,

\* then it is Physical Port object id.

\* If Queue Type is SAI\_QUEUE\_TYPE\_MULTICAST\_VOQ,

\* then it is Switch object id or Null object.

\* If it is a Null object id, then the Queue does not require attachment point. \*/

SAI\_QUEUE\_ATTR\_ATTACHMENT\_ID,

/\* READ-WRITE \*/

…

/\* -- \*/

/\* Custom range base value \*/

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

} sai\_queue\_attr\_t;

## Changes to saiqosmaps.h

/\*\*

\* @brief Enum defining qos map types.

\*/

typedef enum \_sai\_qos\_map\_type\_t

{

…

/\*\* Qos Map to set traffic class to Unicast VOQ \*/

SAI\_QOS\_MAP\_TC\_TO\_UNICAST\_VOQ = 0x0000000e,

/\*\* Qos Map to set traffic class to Multicast VOQ \*/

SAI\_QOS\_MAP\_TC\_TO\_MULTICAST\_VOQ = 0x0000000f,

…

} sai\_qos\_map\_type\_t;

# Configuration Example

TBD