OVSNFV Specs

Release draft (319e5e7)

OPNFV

June 29, 2016

CONTENTS

1 Example Spec - The title of your blueprint **1**
- 1.1 Problem description . . . 1
- 1.2 Proposed change . . . 2
- 1.3 Implementation . . . 3
- 1.4 Dependencies . . . 4
- 1.5 Testing . . . 4
- 1.6 Documentation Impact . . . 4
- 1.7 References . . . 4
- 1.8 History . . . 4

2 High Priority Traffic Path **5**
- 2.1 Problem description . . . 5
- 2.2 Proposed change . . . 6
- 2.3 Implementation . . . 9
- 2.4 Dependencies . . . 9
- 2.5 Testing . . . 9
- 2.6 Documentation Impact . . . 10
- 2.7 References . . . 11
- 2.8 History . . . 11

CHAPTER

ONE

EXAMPLE SPEC - THE TITLE OF YOUR BLUEPRINT

Include the URL of OPNFV wiki page description:

https://wiki.opnfv.org/display/ovsnfv/OVSFV+Requirement+-+Example

Introduction paragraph – why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the process:

- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

Some notes about using this template:

- Your spec should be in ReSTructured text, like this template.
- Please wrap text at 79 columns.
- Please do not delete any of the sections in this template. If you have nothing to say for a whole section, just write: None
- For help with syntax, see http://sphinx-doc.org/rest.html
- To test out your formatting, build the docs using sphinx
- If you would like to provide a diagram with your spec, ascii diagrams are required. http://asciiflow.com/ is a very nice tool to assist with making ascii diagrams. The reason for this is that the tool used to review specs is based purely on plain text. Plain text will allow review to proceed without having to look at additional files which can not be viewed in gerrit. It will also allow inline feedback on the diagram itself.

1.1 Problem description

A detailed description of the problem. What problem is this blueprint addressing?

1.1.1 Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

1.2 Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

If this is one part of a larger effort make it clear where this piece ends. In other words, what's the scope of this effort?

At this point, if you would like to just get feedback on the problem and proposed change, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

1.2.1 Alternatives

What other ways could we do this thing? Why aren't we using those? This doesn't have to be a full literature review, but it should demonstrate that thought has been put into why the proposed solution is an appropriate one.

1.2.2 OVSDB schema impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?

1.2.3 User interface impact

Each user interface that is either added, changed or removed should have the following:

- Specification for the user interface
- Example use case including typical examples for both data supplied by the caller and the response

1.2.4 Security impact

Describe any potential security impact on the system. Some of the items to consider include:

- Does this change touch sensitive data such as tokens, keys, or user data?
- Does this change alter the interface in a way that may impact security, such as a new way to access sensitive information?
- Does this change involve cryptography or hashing?
- Does this change require the use of sudo or any elevated privileges?
- Does this change involve using or parsing user-provided data? This could be directly at the API level or indirectly such as changes to a cache layer.

- Can this change enable a resource exhaustion attack, such as allowing a single interaction to consume significant server resources?

1.2.5 Other end user impact

Aside from the user interfaces, are there other ways a user will interact with this feature?

1.2.6 Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- Will the change include any locking, and if so what considerations are there on holding the lock?

1.2.7 Other deployer impact

Discuss things that will affect how you deploy and configure Open vSwitch that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed? Are the default values ones which will work well in real deployments?
- Is this a change that takes immediate effect after its merged, or is it something that has to be explicitly enabled?
- If this change is a new binary, how would it be deployed?
- Please state anything that those doing continuous deployment, or those upgrading from the previous release, need to be aware of. Also describe any plans to deprecate configuration values or features.

1.2.8 Developer impact

Discuss things that will affect other developers working on Open vSwitch, such as:

1.3 Implementation

1.3.1 Assignee(s)

Who is leading the writing of the code? Or is this a blueprint where you're throwing it out there to see who picks it up?

If more than one person is working on the implementation, please designate the primary author and contact.

Primary assignee: <email address>

Other contributors: <email address>

1.3.2 Work Items

Work items or tasks – break the feature up into the things that need to be done to implement it. Those parts might end up being done by different people, but we're mostly trying to understand the timeline for implementation.

1.4 Dependencies

- If this requires functionality of another project that is not currently used document that fact.
- Does this feature require any new library dependencies or code otherwise not included in Open vSwitch? Or does it depend on a specific version of library?

1.5 Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware.

Please discuss how the change will be tested: Open vSwitch unit tests, VSPERF performance tests, Yardstick tests, etc.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

1.6 Documentation Impact

Which audiences are affected most by this change, and which documentation should be updated because of this change? Don't repeat details discussed above, but reference them here in the context of documentation for multiple audiences. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specification's change.

1.7 References

Please add any useful references here. You are not required to have any reference. Moreover, this specification should still make sense when your references are unavailable. Examples of what you could include are:

- Links to mailing list or IRC discussions
- Links to relevant research, if appropriate
- Related specifications as appropriate
- Anything else you feel it is worthwhile to refer to

1.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand what's happened along the time.

Table 1.1: Revisions

Release Name	Description
2.x	Introduced

CHAPTER

TWO

HIGH PRIORITY TRAFFIC PATH

https://wiki.opnfv.org/display/ovsnfv/OVSFV+Requirement+-+High+Priority+Traffic+Path

2.1 Problem description

A network design may need to adequately accommodate multiple classes of traffic, each class requiring different levels of service in critical network elements.

As a concrete example, a network element managed by a service provider may be handling voice and elastic data traffic. Voice traffic requires that the end-to-end latency and jitter is bounded to some numerical limit (in msec) accuracy in order to ensure sufficient quality-of-service (QoS) for the participants in the voice call. Elastic data traffic does not impose the same demanding requirements on the network (there will be essentially no requirement on jitter. For example, when downloading a large file across the Internet, although the bandwidth requirements may be high there is usually no requirement that the file arrives within a bounded time interval.

Depending on the scheduling algorithms running on the network element, frames belonging to the data traffic may get transmitted before frames belonging to the voice traffic introducing unwanted latency or jitter. Therefore, in order to ensure deterministic latency and jitter characteristics end-to-end, each network element through which the voice traffic traverses must ensure that voice traffic is handled deterministically.

Hardware switches have typically been designed to ensure certain classes of traffic can be scheduled ahead of other classes and are also over-provisioned which further ensures deterministic behavior when handling high priority traffic. However, software switches (which includes virtual switches such as Open vSwitch) may require modification in order to achieve this deterministic behavior.

2.1.1 Use Cases

1. Program classes of service

The End User specifies a number of classes of service. Each class of service will be represented by the value of a particular field in a frame. The class of service determines the priority treatment which flows in the class will receive, while maintaining a relative level of priority for other classes and a default level of treatment for the lowest priority class of service. As such, each class of service will be associated with a priority. The End User will associate classes of service and priorities to ingress ports with the expectation that frames that arrive on these ingress ports will get scheduled following the specified priorities.

Note: Priority treatment of the classes of service cannot cause any one of the classes (even the default class) from being transferred at all. In other words, a strict priority treatment would likely not be successful for serving all classes eventually, and this is a key consideration.

2. Forward high priority network traffic

A remote network element sends traffic to Open vSwitch. The remote network element, indicates the class of service to which this flow of traffic belongs to by modifying a pre-determined but arbitrary field in the frame as specified in Use Case 1. Some examples include the Differentiated Services Code Point (DSCP) in an IP packet or the Priority Code Point (PCP) in an Ethernet frame. The relative priority treatment that frames get processed by Open vSwitch can be guaranteed by the values populated in these fields when the fields are different. If the fields are the same, ordering is not deterministic.

For example: Packet A is sent with a DSCP value of 0 and packet B is sent with a value of 46; 0 has a lower priority than 46. Packet A arrives before packet B. If Open vSwitch has been configured as such, Packet B will be transmitted before Packet A.

2.2 Proposed change

Two possible alternative implementations are outlined below. Further prototyping will be required to determine the favoured solution.

2.2.1 Alternatives

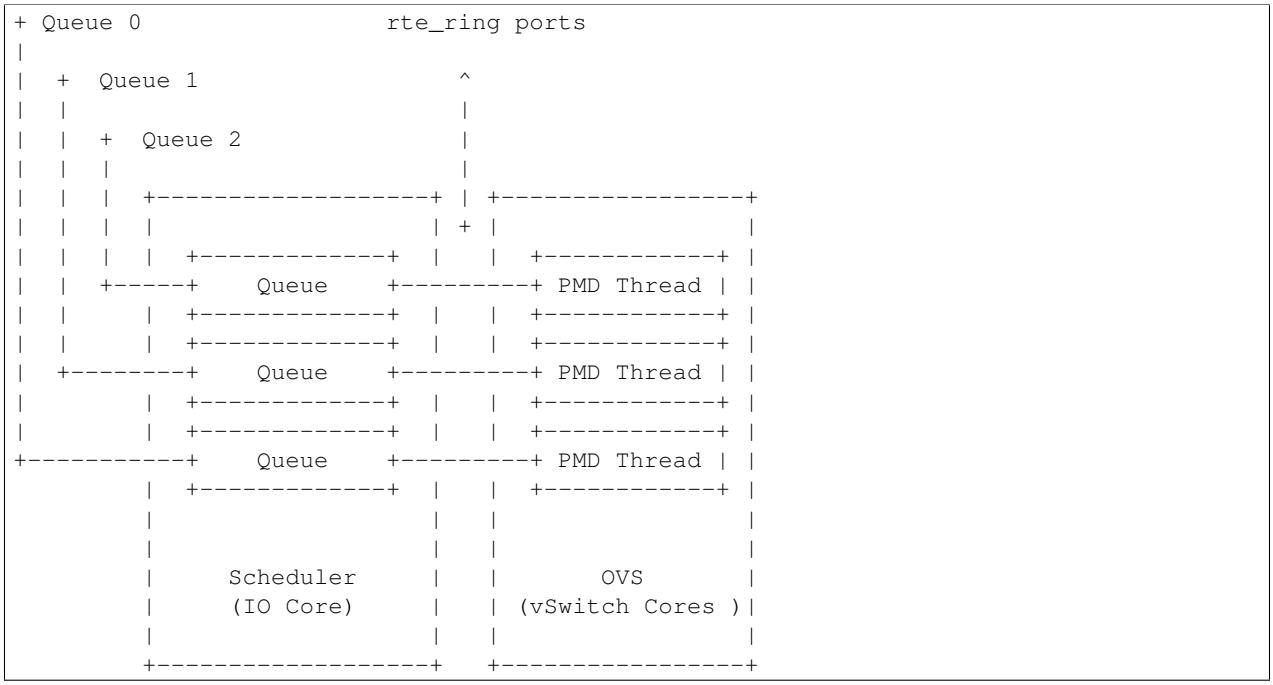
There are currently two alternative implementations for this feature. Further prototyping will be required to determine the proposed change.

Option 1:

Implement an external application that prioritizes traffic before sending to a virtual switch port. This option should not require modifications to OVS but will require more complicated management due the addition of a scheduling application.

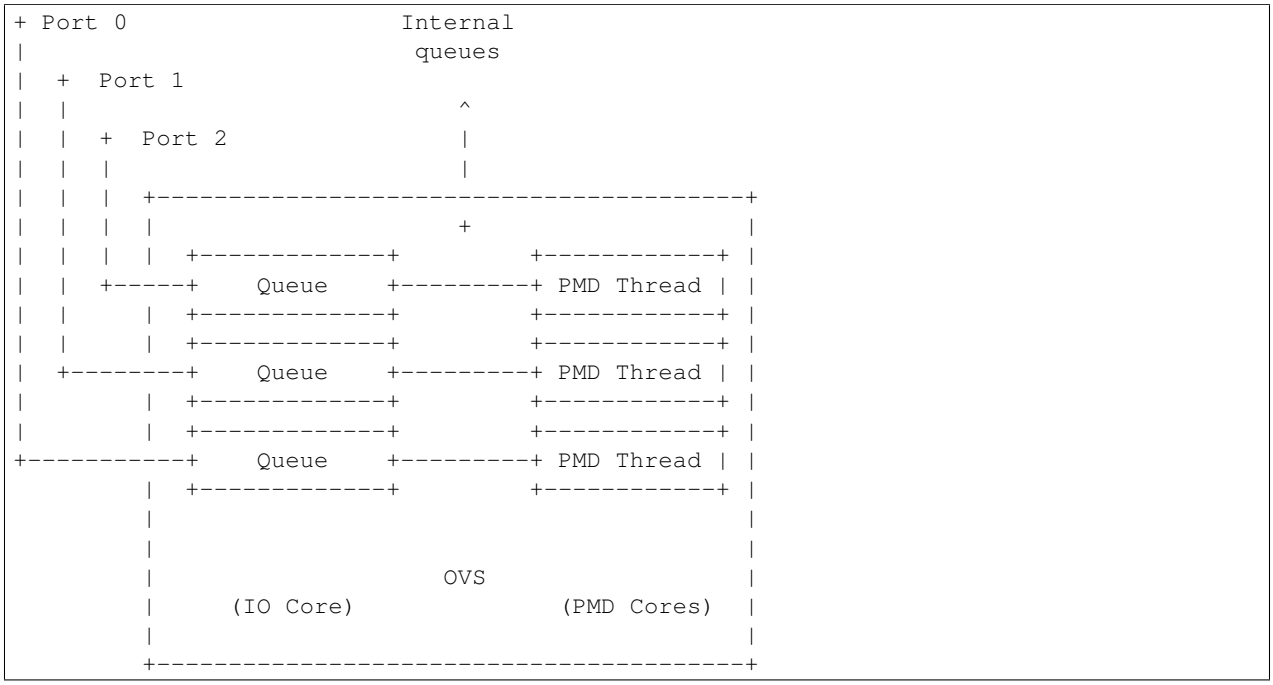
Each OVS ingress queue should have an equivalent ingress queue in the scheduler. The scheduler has responsibility of ordering the frames in it's own queues to ensure they respect the configured priorities.

In this model, OVS receives packets from rte_ring ports that have been provided by the scheduler.

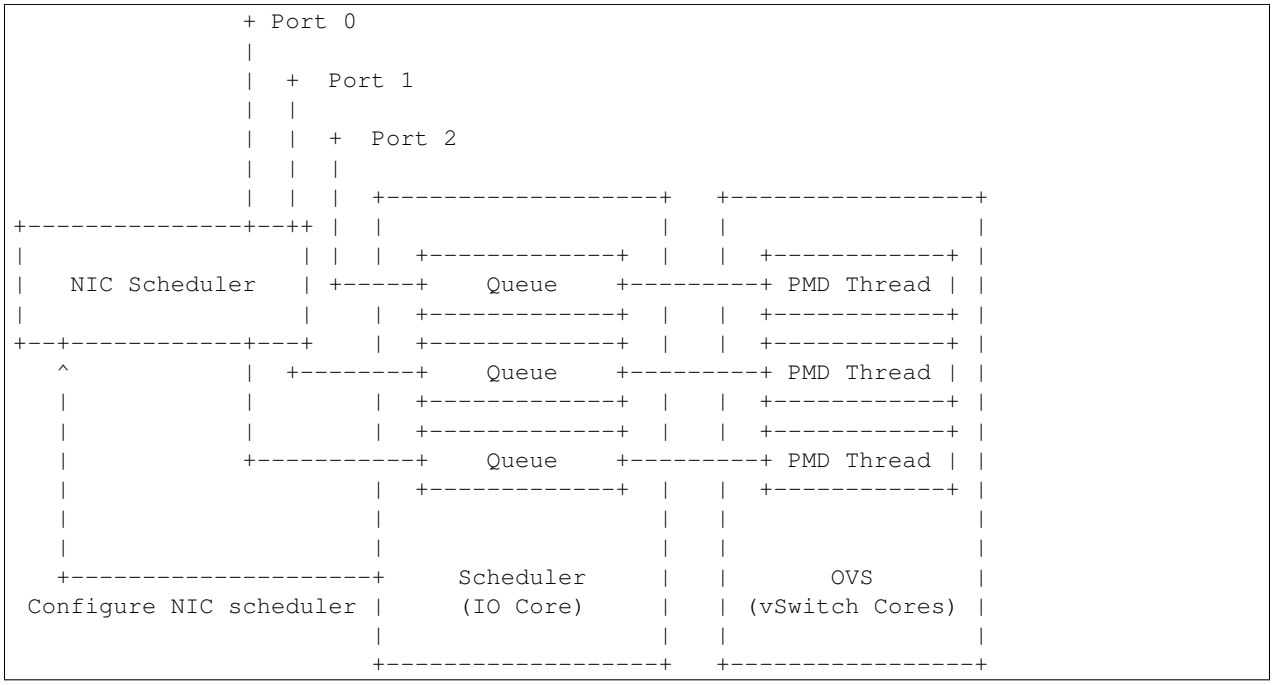


Option 2:

Modify the OVS application to prioritize packets before processing. This will require an IO core in OVS to handle prioritisation of traffic coming from the rx queues in OVS.



It should be noted that for both solutions, it should be possible to offload the scheduling to a capable NIC on ingress. An example of how this could be done for option 1 is seen below.



2.2.2 OVSDB schema impact

As the control interface may be implemented via Open vSwitch, configuration may require updates to the ovsdb schema. An example of how this could be done is presented below:

```
"Classes_of_Service": {
  "columns": {
    "cos_type": {
      "type": "string"},
    "cos": {
      "type": {"key": "integer", "value": "integer", "min": 0, "max": "200"}},
  }
 }
```

2.2.3 User interface impact

A control interface is required which allows the user to:

- Specify the type of field used for determining the class of service. Examples are: dscp, vlan-pcp
- Add a value of that field to a particular priority. Examples are:

```
    Value = 46, Priority = 0
    Value = 0, Priority = 7
```

  It should be possible to specify up to a maximum number (n) of values for each priority. All other traffic is presumed to have the lowest priority. There will be a fixed number of priorities.

- Remove a value from a particular priority
- List priority of a particular class of service

This configuration will be valid for all traffic being handled by the switch.

This control interface may be implemented in via Open vSwitch commands or via an external application (controlling, for example, a NIC or another piece of software).

An example of how this could be controlled via Open vSwitch commands follows:

```
ovs-vsctl add-cos <type> <value> <priority>

ovs-vsctl add-cos dscp 46 0

ovs-vsctl del-cos <type> <value>

ovs-vsctl del-cos dscp 46

ovs-vsctl show-cos 46
```

A similar interface could be used to control an external application.

2.2.4 Security impact

TBD

2.2.5 Other end user impact

TBD

2.2.6 Performance Impact

TBD

2.2.7 Other deployer impact

TBD

2.2.8 Developer impact

TBD

2.3 Implementation

2.3.1 Assignee(s)

Who is leading the writing of the code? Or is this a blueprint where you're throwing it out there to see who picks it up?

If more than one person is working on the implementation, please designate the primary author and contact.

Primary assignee: <email address>

Other contributors: <email address>

2.3.2 Work Items

- Carry out tests to determine current behaviour
- Implement proposed solution alternative 1 as a proof point
-

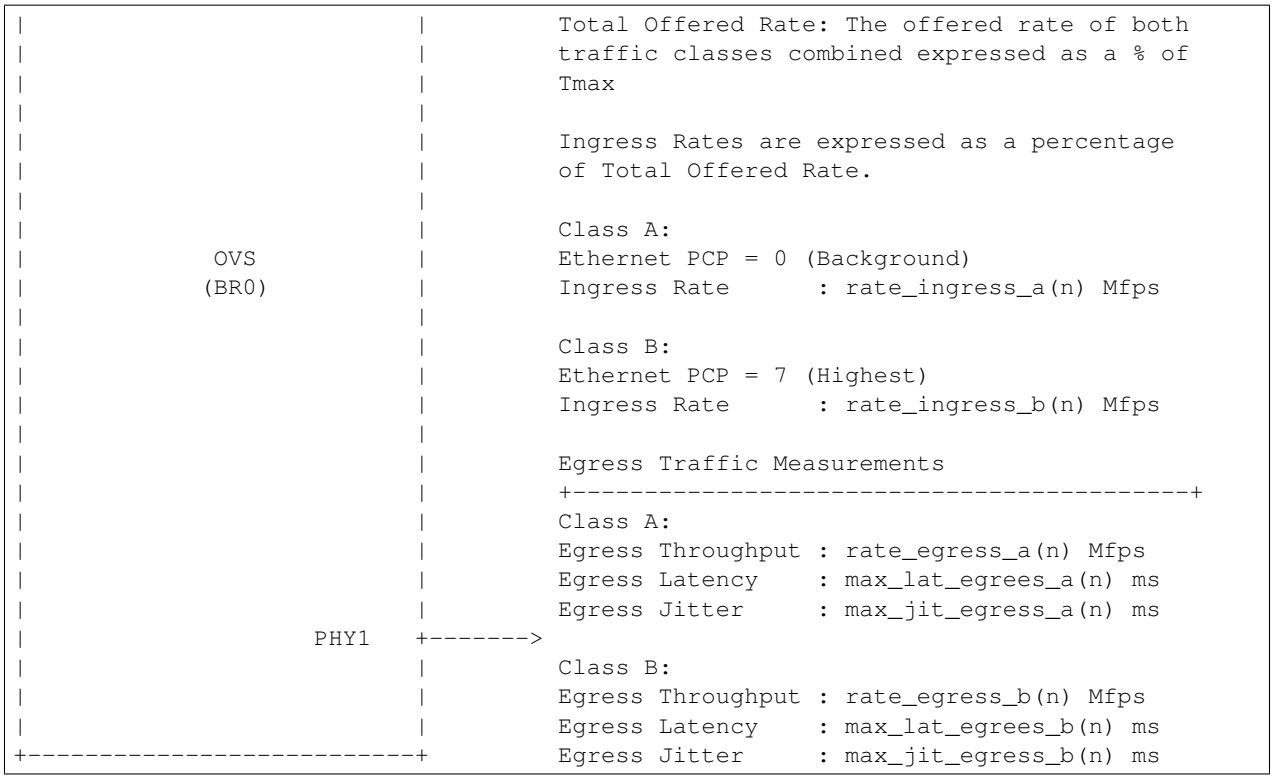
2.4 Dependencies

TBD

2.5 Testing

In order to test how effectively the virtual switch handles high priority traffic types, the following scheme is suggested.:

```
+---------------------------+         Ingress Traffic Parameters
|                           |         +-----------------------------------------+
|                           |
|                           |         Packet Size: The size of the Ethernet frames
|                           |
|                           |         Tmax: RFC2544 Max. Throughput for traffic of
|                     PHY0  <--------+ "Packet Size"
|                           |
```



Open vSwitch is configured to forward traffic between two ports agnostic to the traffic type. For example, using the following command:

ovs-ofctl add-flow br0 in_port=0,actions=output:1

The test will be carried out with the functionality to enable high-priority traffic enabled and disabled in order to guage the change in performance for both cases.

Two classes of traffic will be generated by a traffic generator. In the example above, the classes are differentiated using the Ethernet PCP field. However, another means for differentiating traffic could be used, depending the prioritization scheme that is developed.

Tests should be performed for each combination of:

- Packet Sizes in (64, 512)
- Total Offered Rate in (80, 120, 150)
- rate_ingress_b(n) / rate_ingress_a(n) in (0.1, 0.2, 0.5)

For each set, the following metrics should be collected for each traffic class over a specified time period:

Egress Throughput (Mfps) Maximum Egress Latency (ms) Maximum Egress Jitter (ms)

2.6 Documentation Impact

The following documentation should be updated in OVS

- “man” pages
- NEWS
- INSTALL.DPDK.md

2.7 References

Please add any useful references here. You are not required to have any reference. Moreover, this specification should still make sense when your references are unavailable. Examples of what you could include are:

- Links to mailing list or IRC discussions
- http://lists.opnfv.org/pipermail/opnfv-tech-discuss/2015-December/007193.html
- http://ircbot.wl.linuxfoundation.org/meetings/opnfv-ovsnfv/2016/opnfv-ovsnfv.2016-03-07-13.01.html
- Links to relevant research, if appropriate
- https://wiki.opnfv.org/download/attachments/5046510/qos_mechanisms.pdf?version=1&modificationDate=1459187636000&api=
- Related specifications as appropriate
- Anything else you feel it is worthwhile to refer to

2.8 History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand what's happened along the time.

Table 2.1: Revisions

Release Name	Description
Colorado	Introduced