QoS - Neutron N00bie

Livnat Peer,
Senior Engineering Manager,
Red Hat

Irena Berezovsky,
Senior Architect,
Midokura

David Slama,
Software Director of Cloud and Network Solutions,
Mellanox

Agenda

- Network QoS
- QoS in Neutron
- QoS service design
- Use Case
- Future Work

Network QoS

The ability to guarantee certain network requirements like bandwidth, latency, jitter, and reliability in order to satisfy a Service Level Agreement (SLA) between an application provider and end users.

QoS

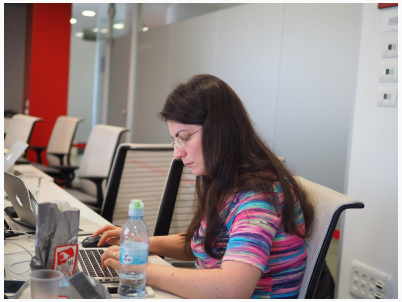
- No industry standard - multiple ways to express bandwidth guarantees
  - OVS - min, max
  - Linux tc - rate, crate, burst, cburst
- Our goals is to enable the cloud administrator to-
  - Control the network resources
  - Tune the network to specific application type
  - Provide different SLAs

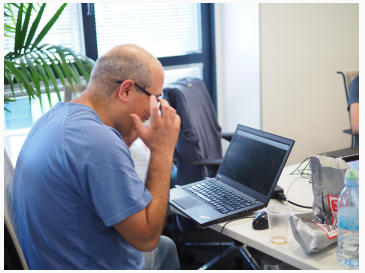
The Noisy Neighbor Problem

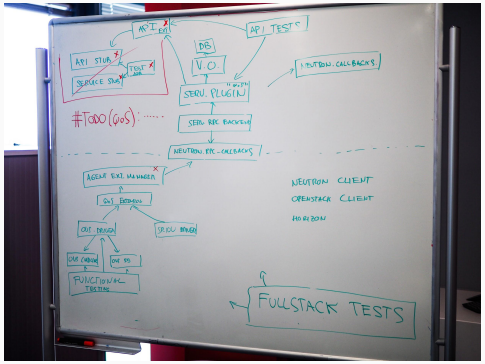
QoS in Neutron - Phase 1

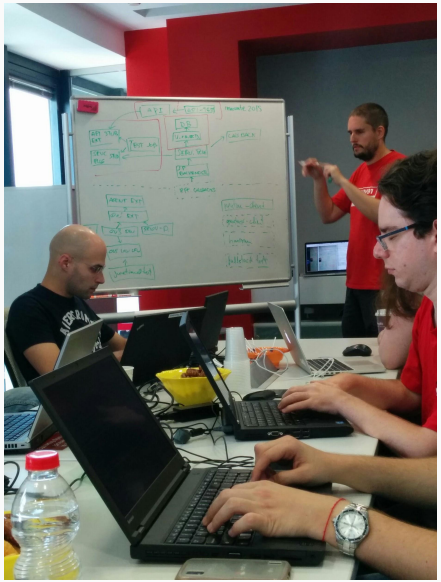
- Adding generic infrastructure that would be extensible for additional use cases
- Scope
  - The current scope was the traffic within the hypervisor
  - Only traffic that leaves the VM (VM-egress)
  - No integration with Nova scheduler

Sprint in Red Hat’s TLV office









QoS API & Data Model - Policy

- A policy is a collection of rules that can be applied on a neutron port
- Policy attributes: Id, Name, Description, Shared, Tenant-Id

```
# neutron qos-policy-create ‘platinum’ \
                --description ‘platinum QoS - charge a lot of $$’
```

- Policy can be associated with Neutron port or network

```
# neutron port-update <port id> --qos-policy ‘platinum’
# neutron net-update <net name> --qos-policy ‘platinum’
```

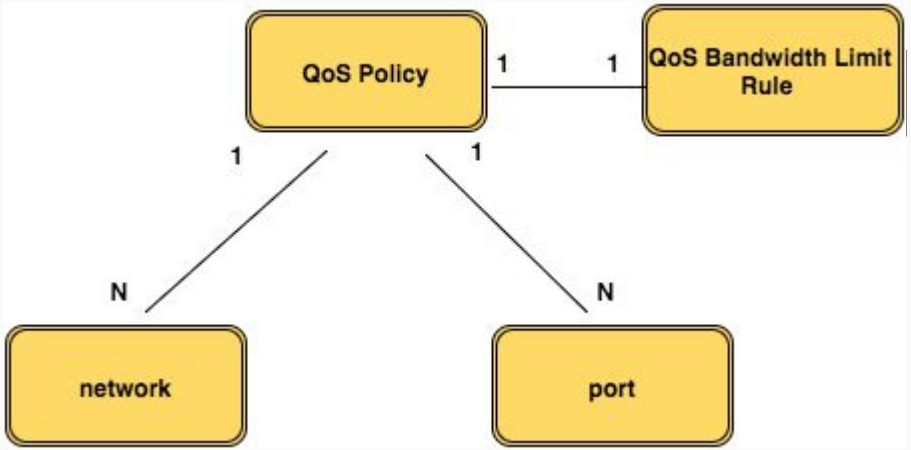
QoS API & Data Model - Rules

- Rule is the building block of a policy
- Abstract QoS Rule
- QoS Bandwidth Limit Rule
  - max-kbps
  - max-burst-kbps

```
# neutron qos-bandwidth-limit-rule-create <policy name> \
        --max-kbps 3000 \
        --max-burst-kbps 300
```

- Future - QoS DSCP Rule
  - dscp-mark

Data Model - Summary



Workflow

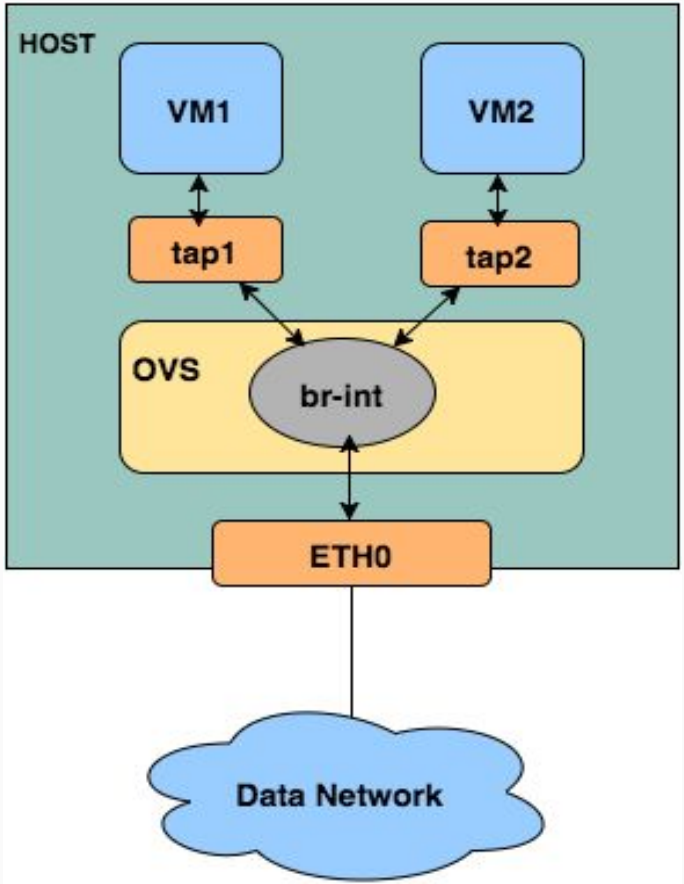
- Typical workflow
  - Creating a policy
  - Adding rules to the policy
  - Associating the policy with a network or a port
- permissions model
  - By default only cloud admin can create a QoS policy
  - Shared vs. non-shared policy
  - The default behaviour can be overridden by changing the policy.json file
- Changes to the Policy immediately propagate to the ports
- Off by default
  - most of the pieces won’t be activated unless explicitly installed, which makes it very low risk of breaking anything for anyone not using QoS

OVS QoS support

VM-ingress == Bridge-egress
VM-egress == Bridge-ingress

Ingress and egress are from the Bridge perspective

- Policing for Ingress Traffic
  - drops packets received in excess of the configured rate
- Shaping for Egress Traffic
  - queues packets received in excess of the configured rate

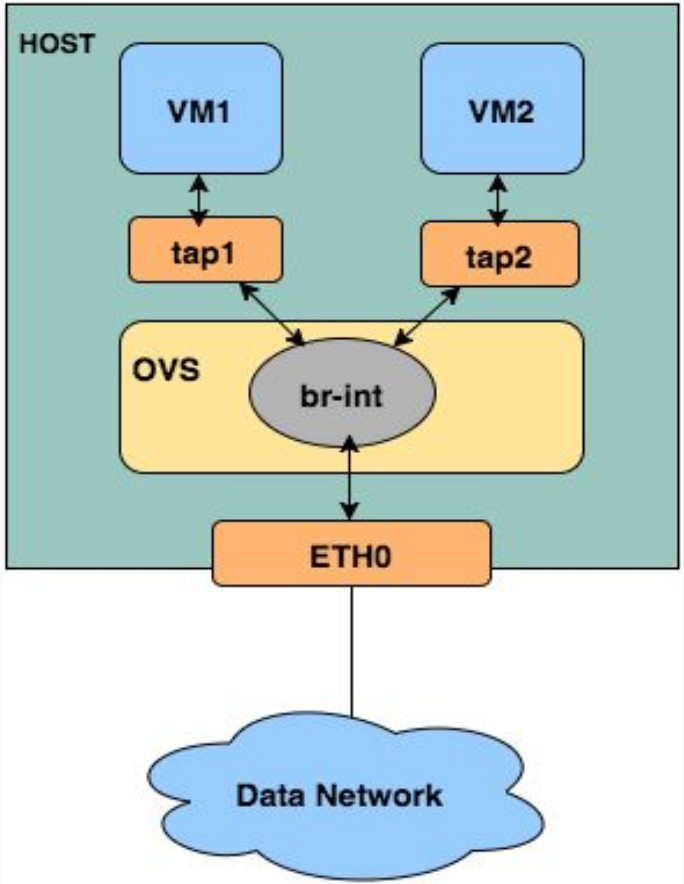


QoS Rate Limit with OVS

- Limit VM egress traffic bandwidth by applying ingress policing settings on OVS port interface

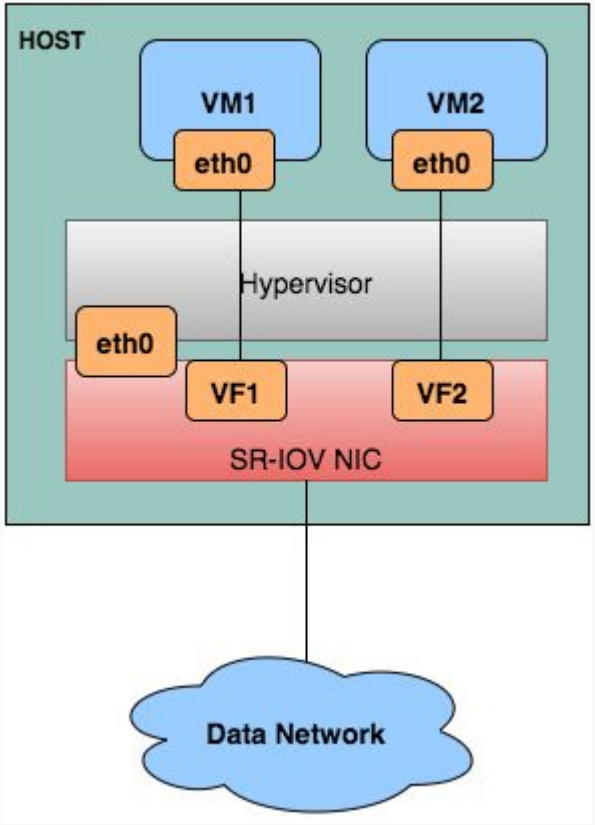
```
# neutron qos-bandwidth-limit-rule-create <policy name> \
        --max-kbps 3000 \
        --max-burst-kbps 300
```

```
# ovs-vsctl set interface tap1 ingress_policing_rate=3000
# ovs-vsctl set interface tap1 ingress_policing_burst=300
```



SR-IOV

- Single Root IO Virtualization - allows a PCIe device to appear as multiple separate PCIe devices (Virtual Functions)
- SR-IOV device can share a single physical port with multiple VMs
- Virtual Functions have near-native performance and provide better performance than para-virtualized drivers and emulated access
- OpenStack supports SR-IOV VF direct passthrough since Juno

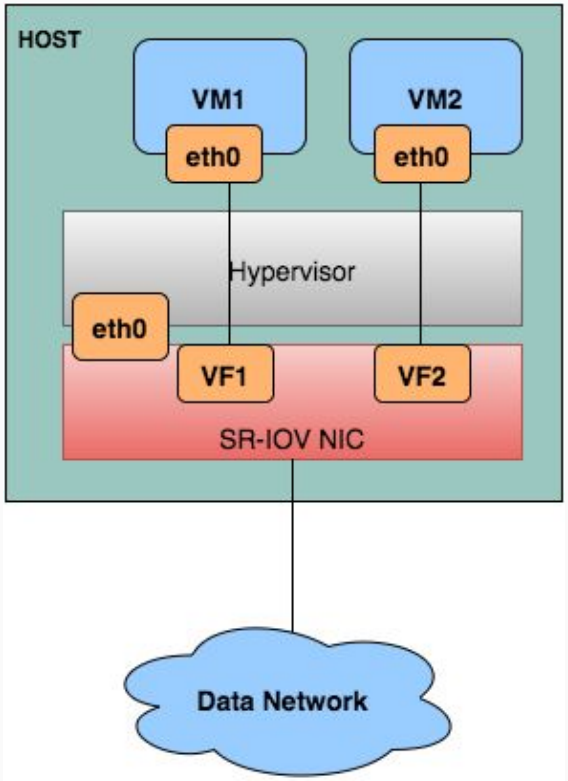


Rate Limit with SR-IOV

- Limit VM egress traffic bandwidth by applying rate limit settings on Virtual Function

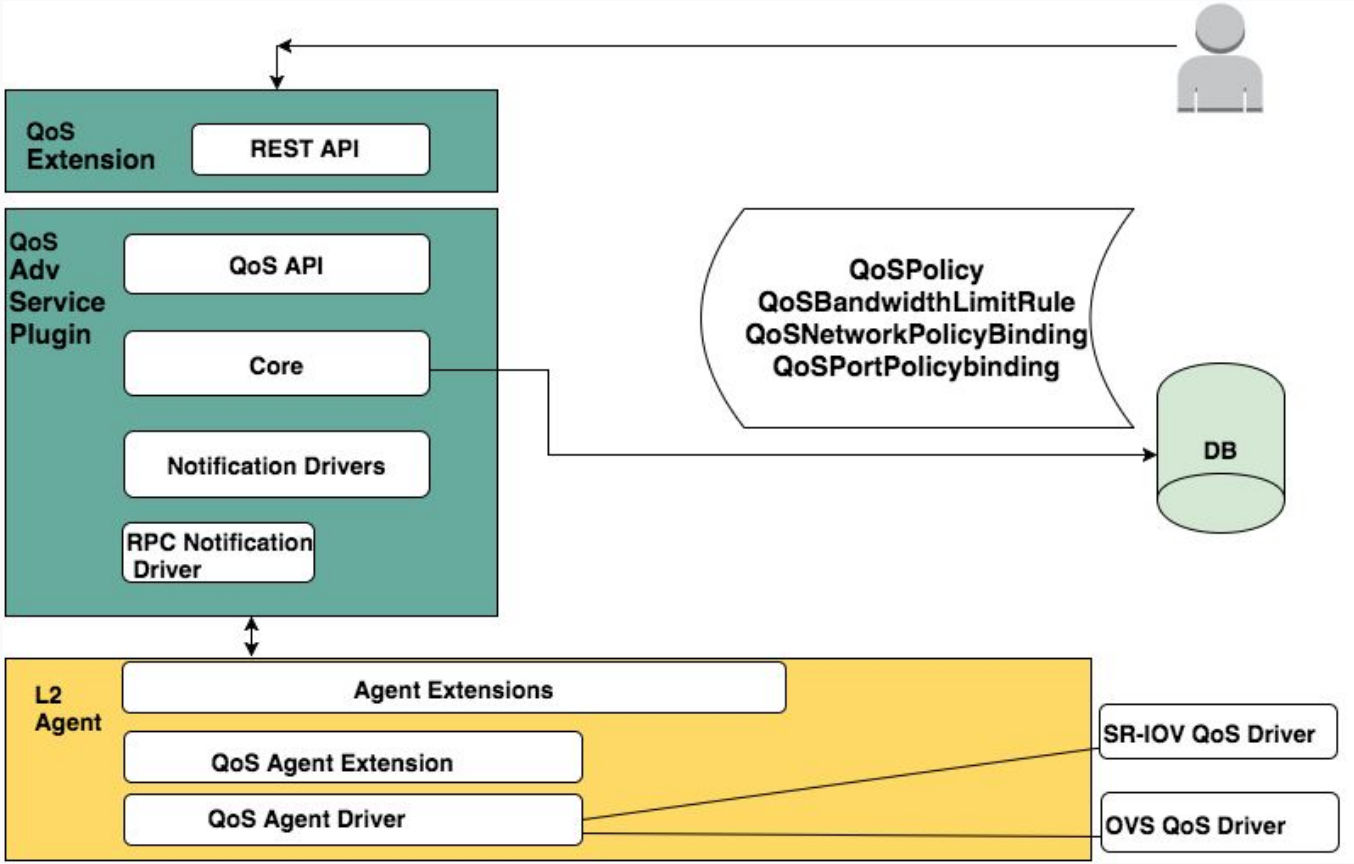
```
# neutron qos-bandwidth-limit-rule-create <policy name> \
        --max-kbps 3000 \
        --max-burst-kbps 300
```

```
# ip link set eth0 vf 1 rate 3
```

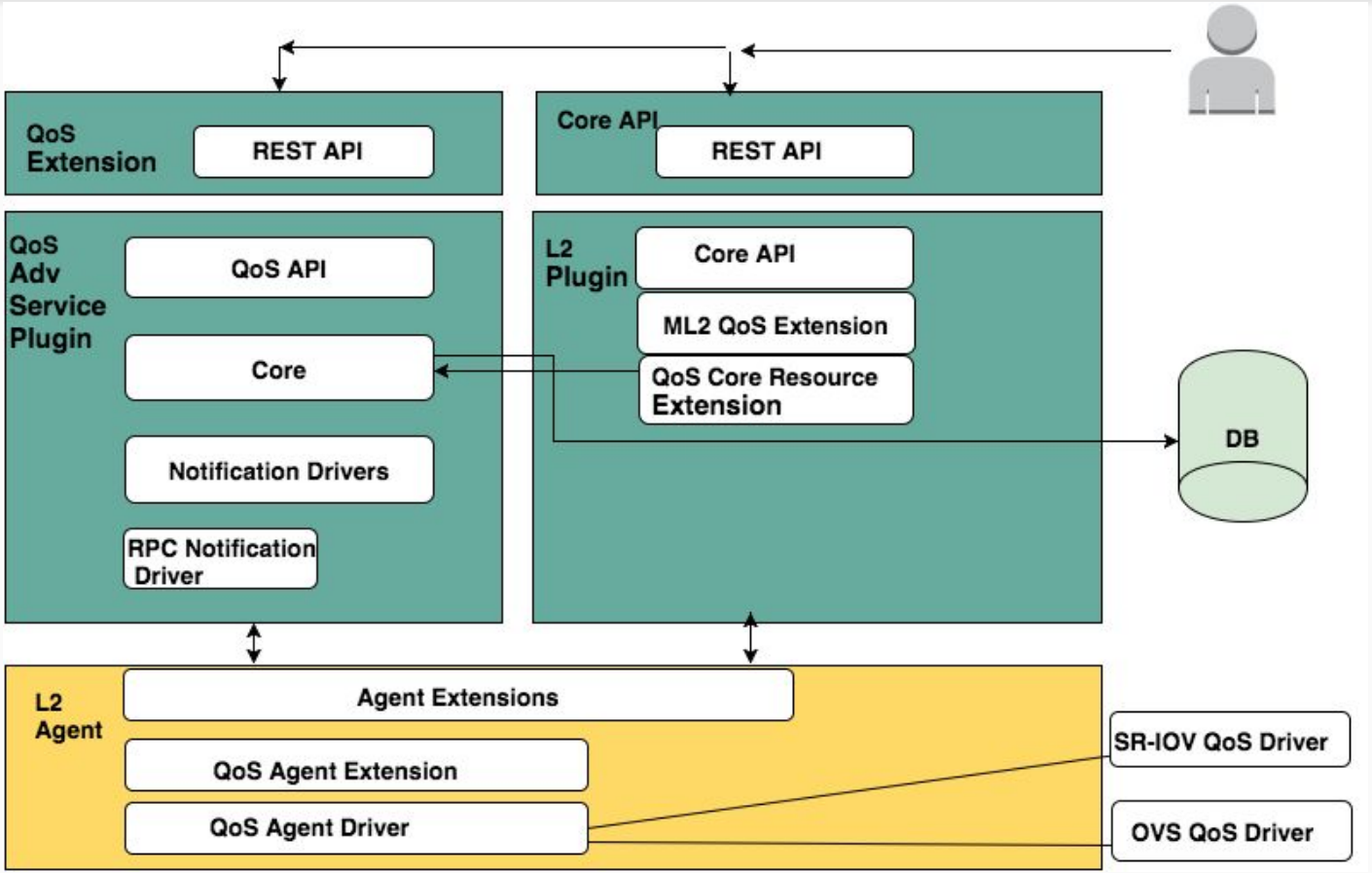



Deep Dive

QoS Neutron Service Design



QoS Neutron Service Design



QoS API Extensibility

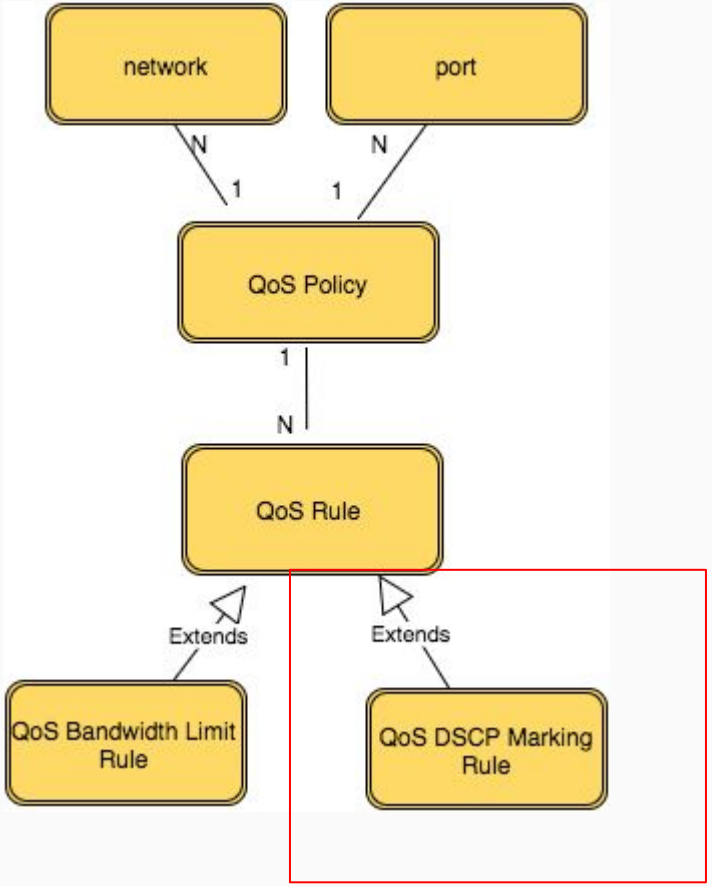
Add QoS Rule Type

Neutron Server

- Define new Rule Type Resource
- Add CRUD methods to QoS Plugin
- Define new DB Model
- Define new versioned object
- Bump QoS Policy version

Neutron Client

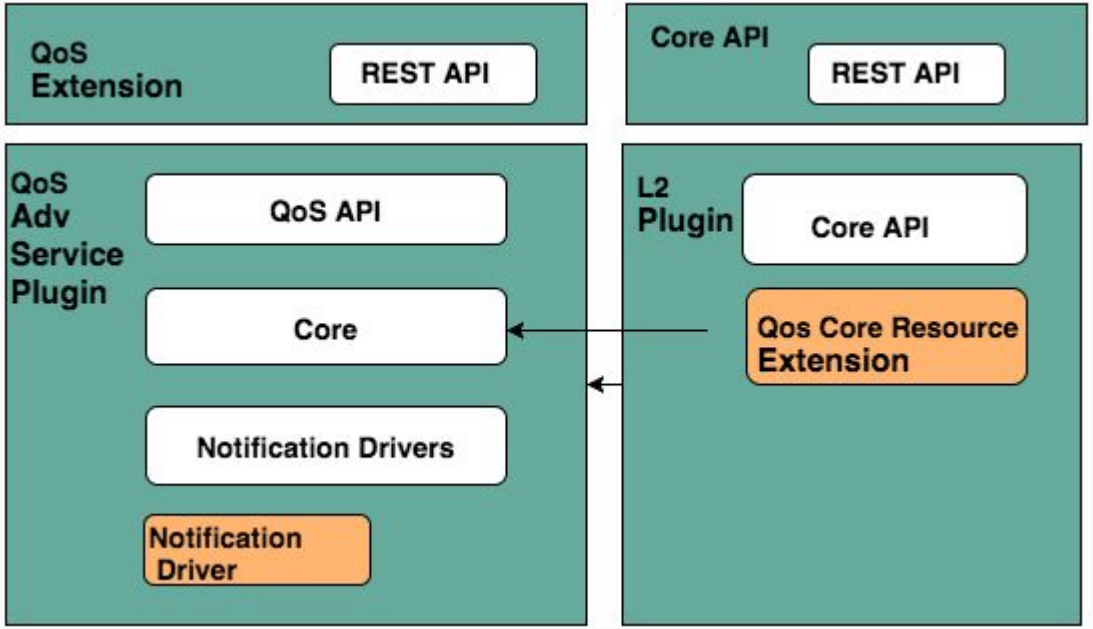
- Add new Rule Type path
- Add CRUD handlers to neutron-cli shell



QoS Service Extensibility

Support QoS API with vendor plugin

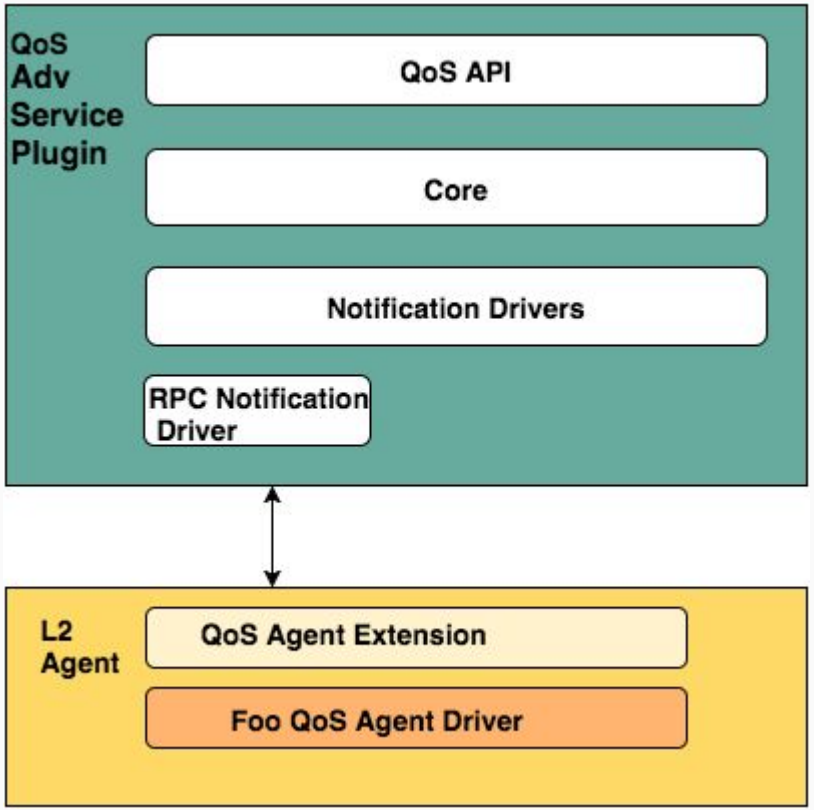
- Declare QoS support rules
- Add new Notification Driver for QoS create / delete / update ops.
- Add QoS Resource Extension to Vendor Plugin to delegate QoS policy port mapping to QoS Advanced Service Plugin



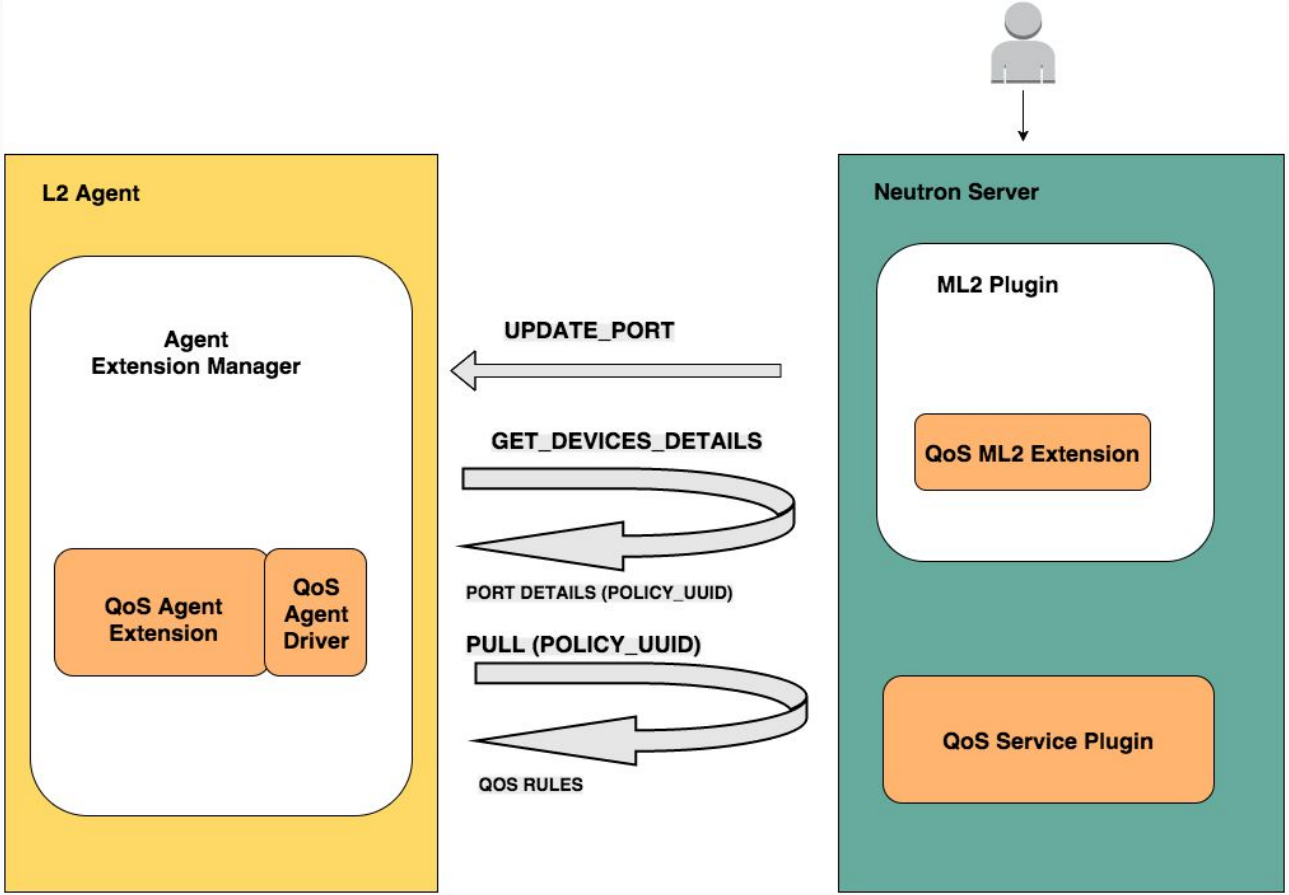
QoS L2 Agent Extensibility

Support QoS with L2 Agent

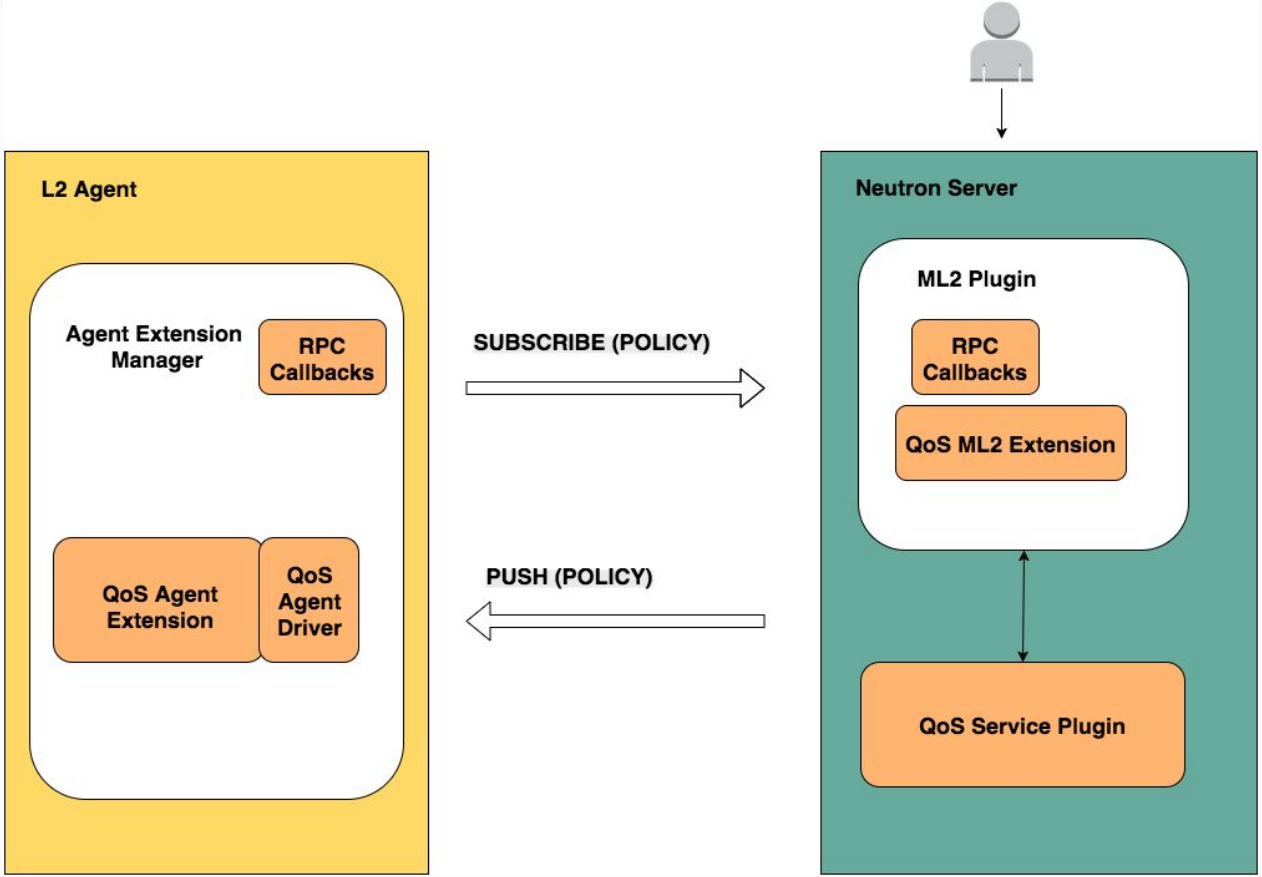
- Add QoS Agent Driver to implement Driver API for L2 Agent managed virtual switch technology



ML2 - Attach QoS Policy



ML2 QoS Policy Update



QoS – Real Life (Customer) Use Case

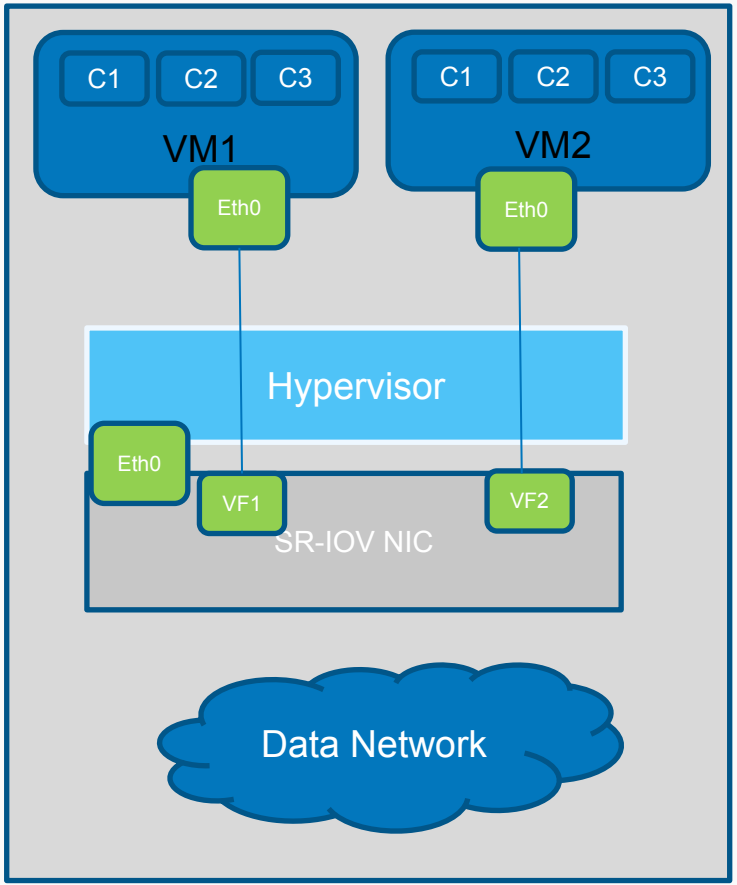
Customer Requirements

- Multiple tenants, each tenant with different QOS requirements
- High Availability for network connectivity
- Each tenant can create one or more containers
- Each container is used to run an application (e.g. VNF)
- Network Auto Provisioning (Segmentation and Policy)
- Option to reflect the QOS settings from the TOR the VM

QoS – Real Life (Customer) Use Case

Host Side

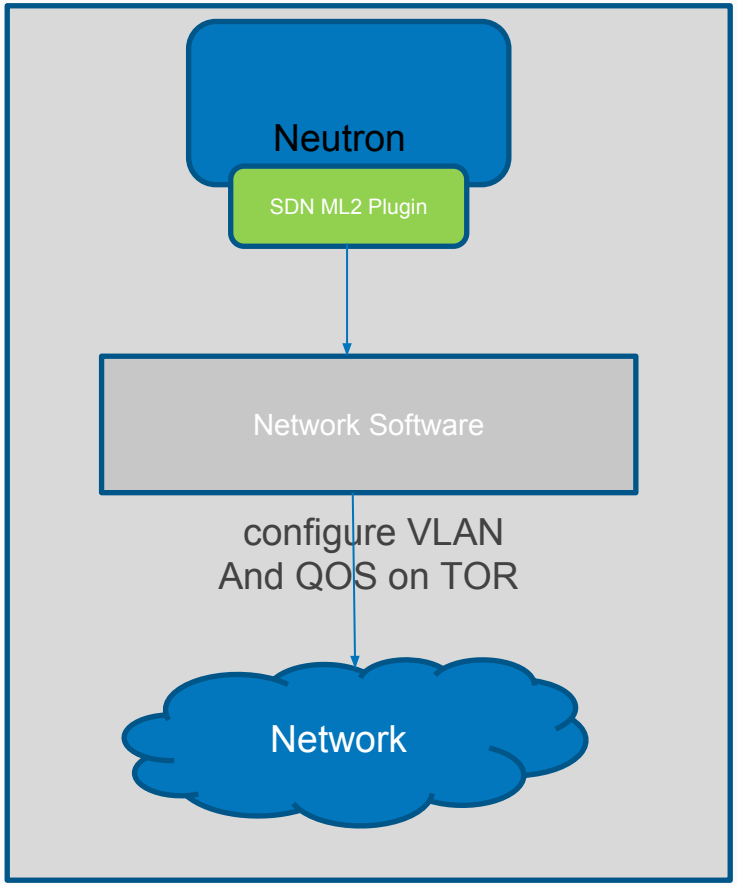
- Multiple tenants, each with a single VM
- Each tenant has multiple applications
- Each application runs in a container
- Each VM – per each tenant, has its own bandwidth share (via rate limiting each VM and ensuring the total is less than the link BW.



QoS – Real Life (Customer) Use Case

Network Side

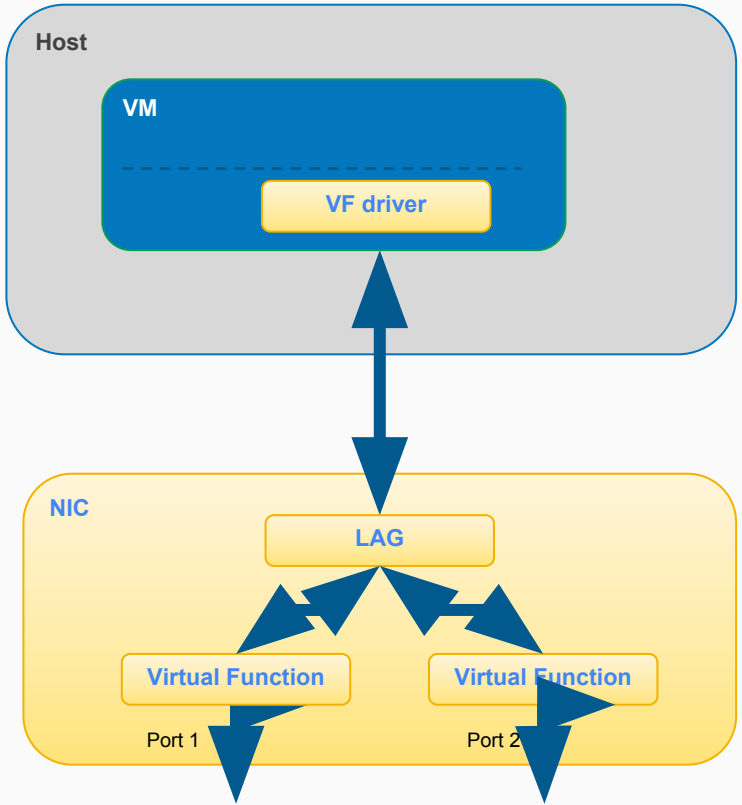
- ML2 SDN Plugin sends data regarding port/network/binding (see next slide)
- ML2 SDN Plugin sends data regarding the Policy (see next slide)
- Reflecting QOS settings on the TOR switch towards the VM



QoS – Real Life (Customer) Use Case

Network Side

- VF LAG for Network HA
- ML2 SDN Plugin sends data regarding port/network/binding
- ML2 SDN Plugin sends data regarding the Policy
- Adding QOS to TOR switch and (ingress policy)



Future Work

- Marking
  - DSCP Marking, https://review.openstack.org/#/c/190285/25/specs/mitaka/ml2-ovs-qos-with-dscp.rst
  - VLAN 802.1p, https://bugs.launchpad.net/neutron/+bug/1505631
  - IPv6 Traffic Class
- Linux Bridge based implementation -
  - https://review.openstack.org/#/c/236210/
- Traffic classifiers
  - https://review.openstack.org/#/c/190463/
- RBAC (Role Based Access Control) integration
- Bandwidth guarantee
  - Nova scheduler integration
- Upgrade - preliminary requirement

Q&A

Resources

- Neutron QoS API Extension - Neutron spec
- Ajo's Blog - Neutron Quality of Service coding sprint
- DSCP Marking - Neutron spec
- Add Classifier Resource - Neutron spec
- User Guide for QoS
- The noisy neighbor problem

Configuration to enable neutron QoS

- On server side
  - enable qos service in service_plugins;
  - set the needed notification_drivers in [qos] section (message_queue is the default);
  - for ml2, add 'qos' to extension_drivers in [ml2] section.
- On L2 agent side
  - add 'qos' to extensions in [agent] section.
- To enable QoS in devstack, update local.conf
  - enable_plugin neutron git://git.openstack.org/openstack/neutron
  - enable_service q-qos

Infra Changes

- Generic RPC Callback
- L2 Agent Extensions Manager & Agent Extensions
- Oslo Versioned Objects
- Core Resource Extensions

QoS – Real Life (Customer) Use Case

Message Example


```
"network_qos_policy": {
"versioned_object.version": "1.0",
"versioned_object.name": "QosPolicy",
"versioned_object.data": {
"description": "",
"rules": [
 {
      "versioned_object.version": "1.0",
      "versioned_object.name": "QosBandwidthLimitRule",
      "versioned_object.data": {
      "max_kbps": 10000,
      "id": "eb48ade9-4a63-4307-acc2-87a31ae68346",
      "max_burst_kbps": 0,
      "qos_policy_id": "7bba8b67-bd58-4370-b524-f58ae4ad50e5"
  },
   "versioned_object.namespace": "versionedobjects"
```