The QueuePusher: Enabling Queue Management in OpenFlow

David Palma[1], Joao Goncalves[1], Bruno Sousa[1], Luis Cordeiro[1],
Paulo Simoes[2], Sachin Sharma[3], Dimitri Staessens[3]
[1]OneSource, Portugal, [2] University of Coimbra, CISUC, Portugal, [3]Ghent University, iMinds, Belgium
E-mail: [1]{palma, john, bmsousa, cordeiro}@onesource.pt,
[2]psimoes@dei.uc.pt, [3]{sachin.sharma, dimitri.staessens}@intec.ugent.be

***Abstract*—The evolution of Software-Defined Networking and the overall acceptance of protocols such as OpenFlow, demonstrates the added value of decoupling the data plane from the control plane. Existing SDN Controllers enable the expected flexibility from such networks by dynamically providing a fine-grained control of each flow. However, hardware-specific configurations, such as the creation of queues or other mechanisms is out of the scope of these controllers. This work presents an extension to a well known OpenFlow controller (Floodlight) to efficiently handle the management of Traffic Control Queues in OpenFlow switches, resorting to a RESTful northbound interface. The obtained results demonstrate further possibility of developing innovative on-demand resource reservation mechanisms in SDN without adding unbearable overheads.**

I. Introduction

Quality of Service (QoS) assurance in networking is indisputably one research topic that has always raised significant challenges. While many solutions have been proposed for typical networks, such as DiffServ [1] or IntServ [2], vendor-specific hardware and management protocols make more complex their overall integration.

An emerging Future Internet technology that introduces a new networking paradigm is Software Defined Networking (SDN), such as OpenFlow (OF) [3], which separates the control plane from the data plane by removing management software from forwarding equipment such as switches or routers. This responsibility is now shifted and embedded into one or more external entities called Controllers.

By using the OpenFlow protocol the enforcing of networking rules can be achieved by exchanging OF messages between the Controllers and the forwarding equipment. This is done per flow by matching a rule or set of rules, allowing also to assign a flow to available traffic shaping queues. The availability of existing network resources can be discovered through the OF standard protocol messages, however the configuration of the network components is not foreseen. In fact, specific OpenFlow Configuration Protocols such as the OpenFlow Management and Configuration protocol (OF-CONFIG) [4] or the The Open vSwitch Database Management Protocol (OVSDB) [5] have been proposed in order to allow the configuration of parameters and resources such as queues. However, these protocols are still not available in existing OF Controllers.

The presented work proposes an implementation of the OVSDB protocol integrated with the northbound API of one of the most well known OF Controllers, Floodlight [6]. An analysis of existing works on the topic of providing QoS in OpenFlow networks is presented in Section II, followed by the proposal of the QueuePusher module (Section III), which enables the dynamic creation of traffic shaping Queues in Open vSwitch [7]. Finally, considerations about the obtained results and future thoughts are presented in Section IV.

II. Quality of Service in OpenFlow Networks

The study of Quality of Service mechanisms in Software-defined Networking scenarios has already been addressed by different authors [8], [9], [10]. However, these works rely on existing Queues or other reserved resources previously configured in the forwarding hardware. While this approach may be considered, it goes against the SDN paradigm and limits its potential. By relying on previous configurations these approaches are able to map Flow Entries to a particular queue, resorting to the enqueue action provided by the OpenFlow protocol. This is specified since OpenFlow specification 1.0 [3], which also allows Controllers to query switches about existing queues.

While QoS can be provided in OpenFlow switches or routers by configuring priority queues and mapping Flow Entries onto these priority queues, to the extent of our knowledge no available controllers provide a standardized management of queues and other works resort to virtualized frameworks [11].

For configuring queues, OpenFlow already describes a central entity, known as the configuration point, responsible for configuring the queues using the OF-Config or OVSDB protocols. The OF-Config protocol is currently being standardized by ONF, however, the OVSDB protocol is already standardized by the Internet Engineering Task Force (IETF). Moreover, the OVSDB protocol is already implemented in Open vSwitch, one of the most commonly used OpenFlow switches, reason why this work focuses on OVSDB.

III. The QueuePusher

The required communication for queue configuration, between an OpenFlow Controller and its corresponding Switches, is not foreseen in the current implementation of the used OpenFlow Switch, the Open vSwitch (OVS). Consequently, we have designed an architecture capable of generating the appropriate queue configuration messages, following the OVSDB standard that is present in OVS.

Since OVSDB is not part of existing OF Controllers, the QueuePusher was created with the intention of providing an interface to be exposed by the well-known Floodlight controller, in order to ease the process of queue creation within

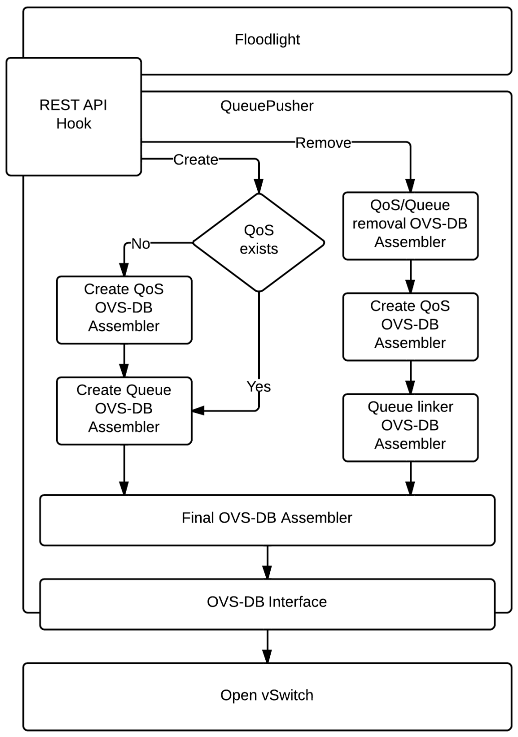
Floodlight
REST API Hook
QueuePusher
Create
Remove
QoS exists
No
Yes
QoS/Queue removal OVS-DB Assembler
Create QoS OVS-DB Assembler
Create Queue OVS-DB Assembler
Create QoS OVS-DB Assembler
Queue linker OVS-DB Assembler
Final OVS-DB Assembler
OVS-DB Interface
Open vSwitch

Fig. 1. QueuePusher Architecture and Operations

our OF enabled switch. The QueuePusher presents itself as an extension to Floodlight, as a module per se, rather than as a standalone application. This approach allows the QueuePusher to be easily embedded in different Controllers and these controllers with the ability of handling the entirety of event processing, avoiding the creation of additional overhead on the controller communication with the forwarding hardware.

A. Architecture

Presently, our architecture is based on a client/server model that follows the OVSDB standard, depicted in Figure 1. The role of the client is played by Floodlight (which, while running the QueuePusher module, is able to issue messages according to the OVSDB standard) whereas the server role is played by an OpenFlow enabled switch that receives the requested commands and configurations.

Following the guidelines for Floodlight module development, the QueuePusher is registered against the controller and implements the appropriate methods to handle internal messages originating from it. To provide an interface to third parties, as other Floodlight modules do, the QueuePusher provides a comprehensive extension to the existing Floodlight REST API [6], registering specific addresses from which different methods can be accessed.

B. Operation and Results

The designed QueuePusher module provides a CRUD (Create, Read, Update, Delete) API, exposed by Floodlight that allows external entities to manage Open vSwitch. Whenever these REST requests are issued, the QueuePusher, with information provided by the Floodlight controller, assembles a new queue-related request and dispatches it to the appropriate OVS enabled switch (according to its Datapath Identifier, DPID), using the OVSDB protocol. All REST requests are responded to, providing the inquiring party with meaningful responses regarding the state and result of their requests.

The overall operations undertaken by the QueuePusher module are depicted in Figure 1, where it is possible to understand the architecture and the steps necessary to configure an OpenFlow Switch.

A preliminary evaluation has demonstrated that this approach is able to handle a large amount of configuration requests in a timely fashion. This evaluation consisted in creating 254 queues simultaneously, which took less than 37*ms*. Moreover, the deletion of queues was also tested, revealing that the consistency checks performed by Open vSwitch are more time consuming, averaging 388*ms* for the removal of 254 queues. These results are presented in Table I, for 200 runs.

TABLE I. QUEUE MANAGEMENT PERFORMANCE

	Average Time for Completion	Standard Deviation
Queue Creation	36,65481481 *(ms)*	3,837278602 *(ms)*
Queue Deletion	387,5219259 *(ms)*	40,1214679 *(ms)*

IV. CONCLUSIONS AND FUTURE WORK

A Queue Management extension to OpenFlow controllers was defined, supporting the OVSDB standard. The obtained results reveal the feasibility of the solution and the chosen paradigm motivates further experiments in additional controllers.

ACKNOWLEDGMENT

This research has received funding from the EU FP7 under agreement n^o 317576 (CityFlow).

REFERENCES

[1] Blake et al., “An architecture for differentiated services,” in *IETF RFC 1633*, Dec 1998. [Online]. Available: http://tools.ietf.org/html/rfc2475

[2] Braden et al., “Integrated services in the internet architecture: An overview,” in *IETF RFC 1633*, June 1994. [Online]. Available: http://tools.ietf.org/html/rfc1633

[3] Plaff et al., “Openflow switch specication,” Feb 2011. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf

[4] Bansal et al., “Openflow management and configuration protocol,” Apr 2014. [Online]. Available: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow-config

[5] Pfaff et al., “The open vswitch database management protocol,” in *IETF RFC 7047*, Dec 2013. [Online]. Available: http://tools.ietf.org/html/rfc7047

[6] Project Floodlight, “Floodlight,” Jul 2014. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[7] OpenSource Community, “Open virtual switch,” Jul 2014. [Online]. Available: http://openvswitch.org

[8] Sonkoly et al., “On qos support to ofelia and openflow,” in *Software Defined Networking (EWSDN), 2012 European Workshop on*, Oct 2012, pp. 109–113.

[9] Bueno et al., “An opennaas based sdn framework for dynamic qos control,” in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, Nov 2013, pp. 1–7.

[10] Bari et al., “Policycop: An autonomic qos policy enforcement framework for software defined networks,” in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, Nov 2013, pp. 1–7.

[11] Sonkoly et al., “Openflow virtualization framework with advanced capabilities,” in *Software Defined Networking (EWSDN), 2012 European Workshop on*, Oct 2012, pp. 18–23.