Experiences Implementing QoS Aware Routing on Off-the-shelf SDN Switches

Jannis Ohms, Olaf Gebauer,
Nadiia Kotelnikova, Marina Arikova and Diederich Wermser
Research Group IP-Based Communication Systems, Ostfalia University of Applied Sciences,
Salzdahlumer Str. 46/48, D-38302, Wolfenbüttel
jannis.ohms2@ostfalia.de, ola.gebauer@ostfalia.de, nadiia.kotelnikova@hs-anhalt.de, m.arikova@ostfalia.de,
d.wermser@ostfalia.de

Keywords: SDN, OpenFlow, Routing, QoS

Abstract: This paper provides an overview of the Quality of Service (QoS) capabilities defined in the OpenFlow specification. Several vendor documentations from off-the-shelf products are compared with the OpenFlow specification. This research reveals inconsistencies between the specification and the vendors implementation. Queues for examples are not implemented by all vendors. This gap can lead to interoperability problems in a network while using hardware from different vendors. The research also shows, that the majority of vendors provide a port statistic function which gives information about incoming and outgoing bandwidth about each port of a switch. Based on this function a QoS aware routing application for off-the-shelf switches is proposed. This concept can be used to change the flow of traffic in an OpenFlow network based on the utilization of the interfaces. Based on the conducted research, the application can be used with hardware from multiple vendors. This paper does not contain a quantitative evaluation of the implemented application.

1 INTRODUCTION

Software-Defined Networking (SDN) is a new concept for the implementation of computer networks. The most commonly used realization of the SDN concept is the OpenFlow Protocol [1] which is specified by the Open Networking Foundation (ONF). SDN Networks consist out of multiple SDN switches and at least one SDN controller. The controller has an overview of the whole topology and creates flow rules for the switches. The controlled switches forward packets according to the flow rules they received. The use of SDN in large networks creates a requirement for QoS aware routing. The goal of QoS aware routing is to choose one of the multiple possible paths in the network under consideration of the QoS requirements of the transported traffic. Classical routing algorithms were developed for autonomous systems (AS) [2] where each router makes its own routing decisions. In SDN the routing decisions are provided by the centralized SDN controller. This form of centralized control is visualized in Figure 1 and Figure 2. The new form of centralized control makes it necessary to rethink QoS aware routing in the context of OpenFlow. In this paper, the authors will take a look at the QoS mechanisms provided by the OpenFlow specification, and compare it with the QoS mechanisms implemented in commercially available off-the-shelf OpenFlow hardware. This paper also describes a concept to provide QoS aware routing which uses the QoS mechanisms implemented in commercially available off-the-shelf OpenFlow hardware. The concept has been implemented as an SDN application.

2 RELATED WORK

OHMS et al. [3] showed that it is possible to use the queueing mechanism of an off-the-shelf OpenFlow switch to provide QoS for Voice over IP (VoIP) streams. The used topology consists of a single switch. GUCK et al. [4] analysed a set of routing algorithms in the context of SDN. The analysis focused on resource consumption and efficiency. The algorithms have only been simulated and not im-

plemented on top of real OpenFlow hardware. ZHANG et al. [5] compared the performance of OpenFlow with routing protocols like Open Shortest Path First (OSPF). Their results show that OpenFlow can react much quicker to topology changes when compared to OSPF. JINYAO et al. [6] proposed a QoS aware routing concept which uses queueing and queue statistics. The concept has been imple-mented and tested on a single computer using a network simulator. The related work indicates that QoS aware routing is a research topic in the scientific community. Our focus on off-the-shelf OpenFlow hardware makes this paper unique when compared to the related work.

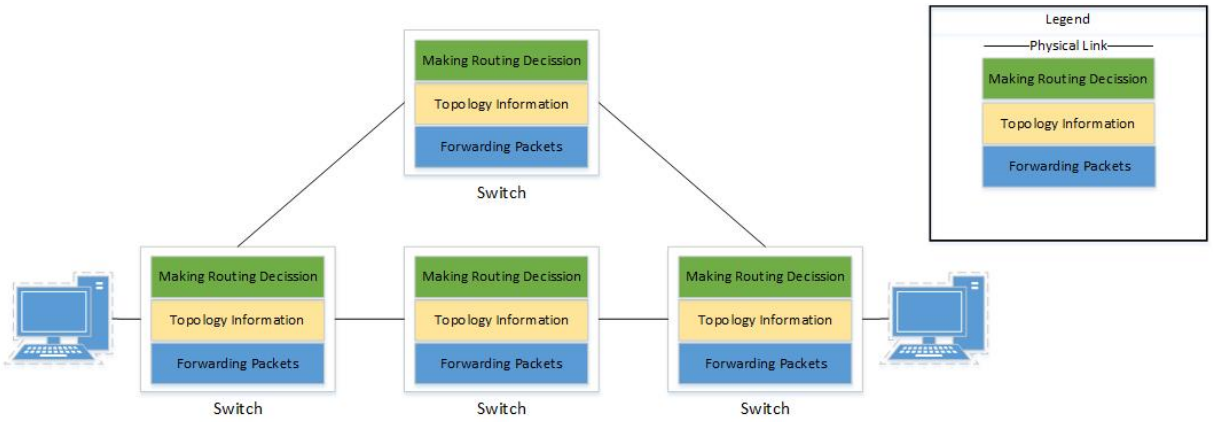

Figure 1: Routing in a classical network.

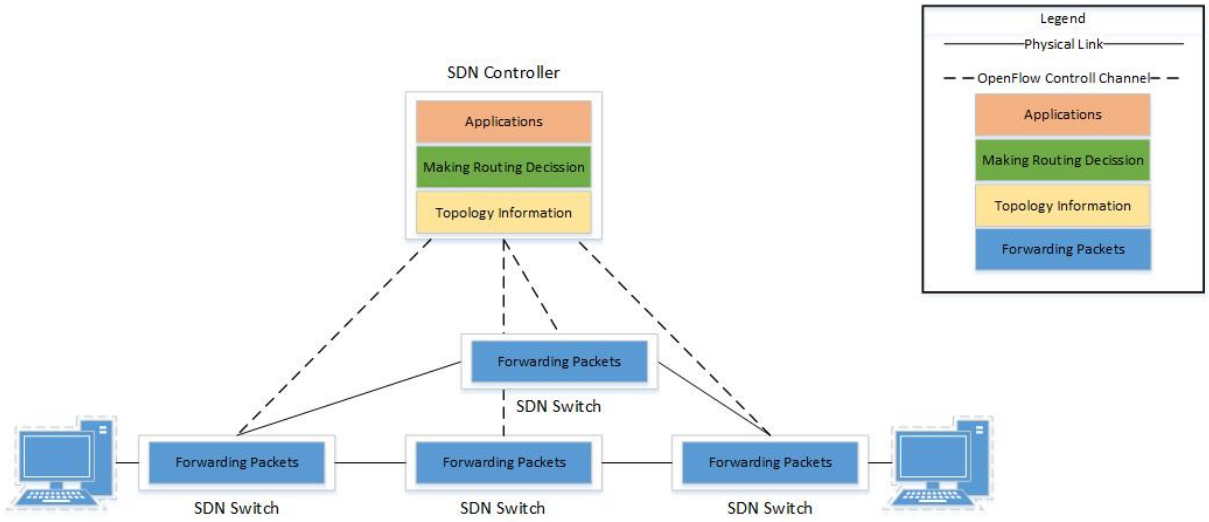

Figure 2: Routing in an SDN network.

3 QOS CAPABILITIES DEFINED IN THE OPENFLOW SPECIFICATION

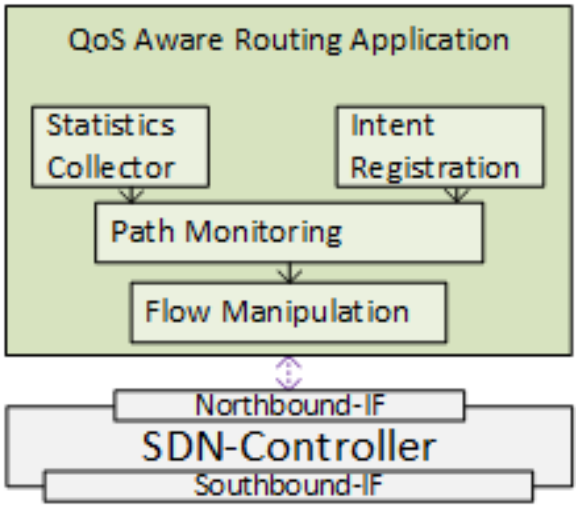

Figure 1: Architecture of the developed QoS aware routing application.

All statements made are based on the OpenFlow specification 1.3 which is the most commonly implemented version regarding the hardware available on the market [1]. OpenFlow 1.3 provides three mechanisms to enable QoS.

Queueing

OpenFlow 1.3 provides a class-based queueing algorithm on the egress port, as a not mandatory part of the specification. The queues provide a guaranteed bandwidth and a maximum bandwidth. The standard specifies OpenFlow messages to configure queues and to get statistics about bandwidth and errors for each queue. Each flow table entry can use an enqueue action to insert a matched packet into a queue. The standard does not specify a precedence between the queues or a required number of queues. The standard also does not specify any queueing algorithm.

Metering

The OpenFlow specification provides a metering mechanism which limits the bandwidth of a given flow. When the limit of a meter is exceeded, all traffic which goes through the meter gets dropped. Alternatively, the Type of Service (ToS) bits of the IPv4 header can be rewritten. The specification contains OpenFlow messages to configure meters and to get statistics about bandwidth and errors for each meter. Each flow table entry can use an action to assign a matched packet in a given meter. The specification does not specify the number of meters which have to be implemented on the switch.

Port Statistics

The specification provides port statistics which measure the sending and receiving bandwidth for each port of the switch. The OpenFlow specification contains a message which enables the controller to collect port statistics.

4 QOS CAPABILITIES OF THE OPENFLOW IMPLEMENTATION FROM DIFFERENT VENDORS

The authors compared the OpenFlow capable product families of four different vendors. Namely HP, Brocade, Juniper, and Pica8. This comparison focuses on the QoS capabilities described in chapter 3. This comparison is based on documentation provided by the vendors [10][11][12][13].

Queueing

Not every vendor implements queues. If they do, they use between 4 to 8 queues per port with a hierarchical precedence. The queue configuration messages are not implemented by any vendor, the queues are usually configured over proprietary CLI interfaces. Most implementations use the HTB scheduling algorithm [7]. The queue statistic messages are only implemented on the Brocade Netiron switches.

Metering

Metering is implemented by all vendors. The meter statistics are not always implemented. The meters are configured through a proprietary CLI interface.

Port Statistics

Port statistics are implemented by all hardware vendors used in this comparison.

Based on this results port statistics seem to be the only commonly available mechanism for our application. The use of port statistics has a significant drawback. The application cannot detect full queues. This can lead to packet loss if a port seems to be idle regarding overall bandwidth consumption while one or more queues exceed their upper bandwidth limit. This problem can be solved using queue statistics.

5 CONCEPT AND ARCHITECTURE OF A QOS AWARE ROUTING APPLICATION

The application has been developed as an internal module for the open source SDN controller Project Floodlight [8]. The architecture of the SDN controller is visualized in Figure 5. The controller provides basic functionality implemented as modules, for example, information about connected hosts, current network topology, etc. Some functions are used for the development of the application. The application for QoS aware routing consists of four logical components.

Statistics Collector

The Statistics Collector collects the port based statistics from the switches. This component contains a background task which gets executed periodically at a fixed rate.

Intent Registration

The Intent Registration component allows the registration of intents with QoS requirements. An intent is a desire of a terminal endpoint to communicate with another terminal endpoint. It consists of a set of header fields which identifies a set of packets. The Intent registration uses the Routing Service of the SDN controller to find every possible path to fulfil the desired intent. All paths get observed by the Path Monitoring.

Path Monitoring

The Path Monitoring component has a background task which receives data from the Statistics Collector to evaluate the bandwidth consumption for every possible port on every path for every registered intent. If the bandwidth capacity of a path which is currently in use by an intent gets exceeded, the Path Monitoring uses an alternative path if possible. This component uses the Flow Manipulation component to change the flow of traffic. The evaluation of every path might not be possible in large topologies. In this case, a preselection is necessary to reduce the set of paths to a size which can be handled by the application. When an intent gets unregistered, the monitoring of the possible paths is canceld.

Flow Manipulation

This Flow Manipulation component uses the Switch Service of the SDN controller to push flow table entries for every switch on a given path. This component opens the possibility to create end-to-end flows for a given path. There is no need to manipulate each flow table of every switch directly.

This architecture is visualized in Figure 3. The routing process can be separated into the following steps.

- If a new intent gets registered, the application looks up every possible path which connects the terminal endpoints of the intent.
- If a path has been found, the application assigns one path to the intent.
- Every packet which matches the header fields (which are specified in the intent) gets forwarded through the assigned path.

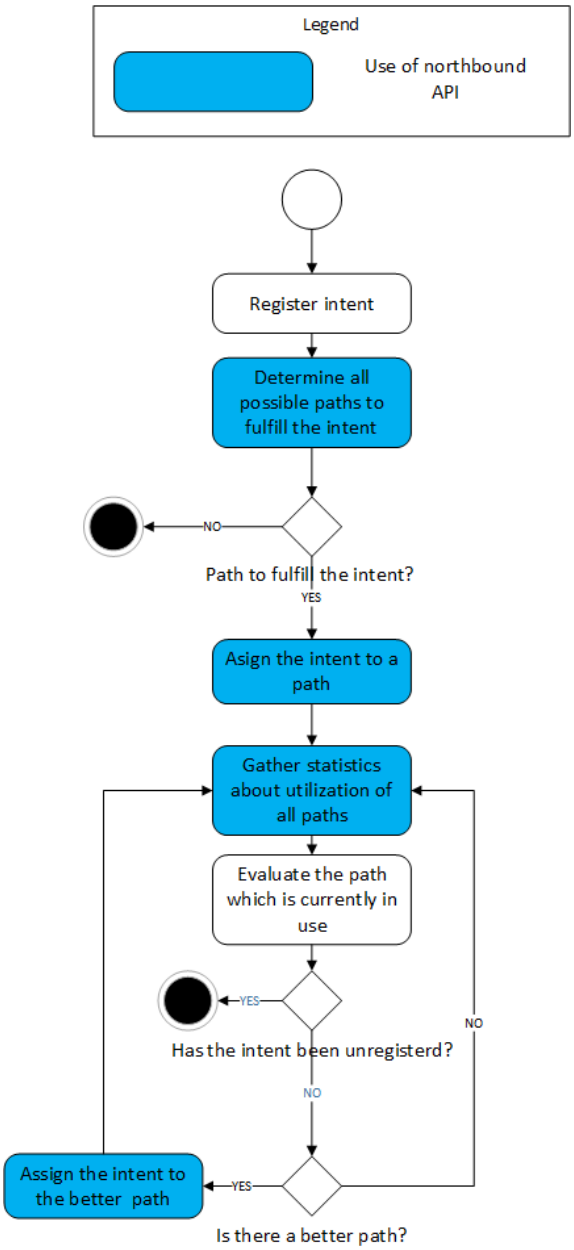

Figure 2: Concept for QoS aware routing presented by the authors.

- The application collects statistical data about the utilization of every port on every path which is currently tracked.
- The statistical data is used to compare the utilization of the current path with every possible path.
- If a better path is available, the application reassigns the intent to the better path.

The concept of the application is visualized in Figure 4.

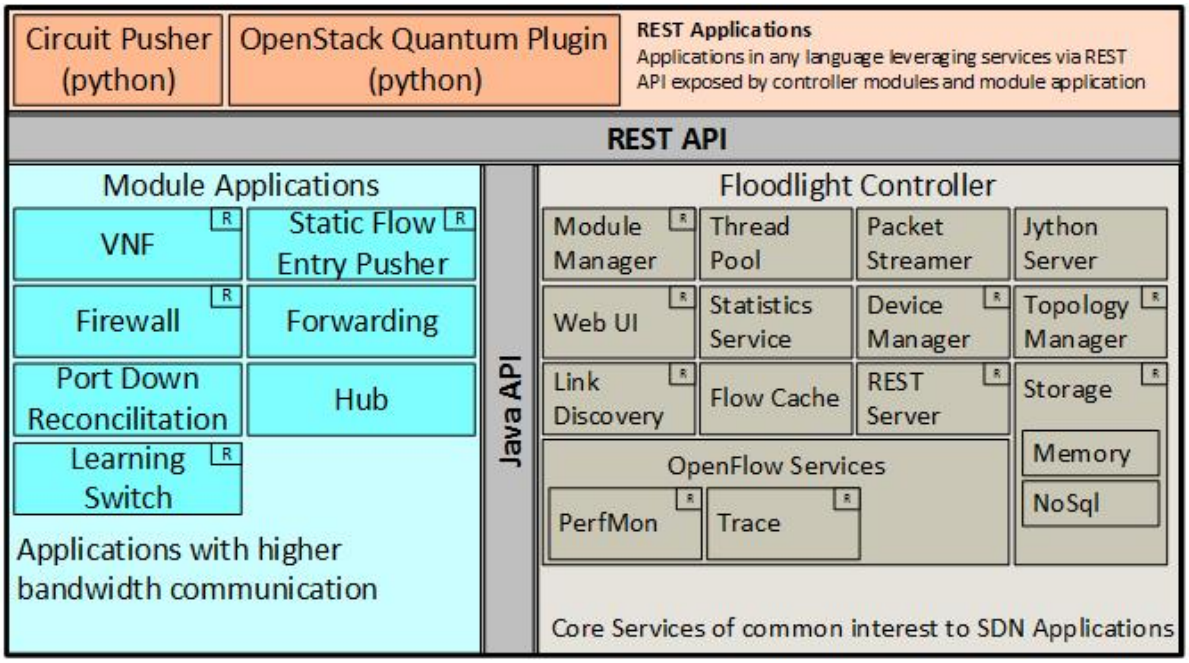

Figure 3: Architecture of the Floodlight SDN Controller.

Table 1: Comparison of QoS capabilities of OpenFlow product families of different vendors.

	Pica8	Juniper	Brocade	HP
Queues	Yes	Yes	Yes	No
Metering	Yes	Yes	Yes	Yes
Queue Statistics	No	No	Not in all products	No
Port Statistic	Yes	Yes	Yes	Yes

6 CONCLUSIONS

There are inconsistencies between features of the OpenFlow specification and the vendor implementations of hardware switches (see Table 1). Depending on the vendor this gap can cause interoperability problems. The ONF specifies new OpenFlow versions every year, which results in outdated hardware. Most OpenFlow features are implemented in the application specific integrated circuit (ASIC) of a switch. In case of a new performance demanding OpenFlow functionality (e.g. queueing), new ASICs and switches have to be developed and deployed. This results in skipping certain version and features. An alternative way to implement new features in existing hardware is the use of P4 [9]. This is a domain specific language which enables software-based packet processing. P4 programs are compiled into hardware. This turns the static switch ASIC into a chip which can be dynamically reprogrammed after it has been deployed as part of a switch. This can be compared to a Field Programmable Gate Array (FPGA). By using P4, vendors can update their hardware after it has been deployed. The proposed concept shows that the routing behaviour of an OpenFlow network can be changed based on the utilisation of the hardware interfaces. A quantitative evaluation of the application is not within the scope of this paper. The proof-of-concept implementation uses the current

bandwidth utilisation to determine the quality of a given path. In future more parameters and models should be evaluated in order to provide guaranteed QoS. This is indispensable in the context of Industry 4.0 and wide area SDN networks.

ACKNOWLEDGMENTS

The research presented in this paper is funded by the BMWi (Bundesministerium für Wirtschaft und Energie) within the ZIM-Program (Zentrales Innovationsprogramm Mittelstand), project INAASCA (Integrated Network as a Service Solution as Part of Cloud IT Application Portfolio) [14]. Additionally this work was supported by the Ministry for Science and Culture of Lower Saxony as part of SecuRIn (VWZN3224), which is funded by the funding initiative "Niedersächsisches Vorab" of Lower Saxony.

REFERENCES

[1] Open Networking Foundation, „OpenFlow Switch Specification Version 1.3.5 (Protocol version 0x04)," [Online] Available: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.3.5.pdf.

[2] A. S. Tanenbaum und D. Wetherall, Computer networks, Prentice hall, 1996.

[3] J. Ohms, O. Gebauer, N. Kotelnikova, D. Wermser und E. Siemens, "Providing of QoS-Enabled Flows in SDN Exemplified by VoIP Traffic," 5th International Conference on Applied Innovations in IT, 2017.

[4] J. W. Guck, A. van Bemten, M. Reisslein und W. Kellerer, "Unicast QoS routing algorithms for SDN," IEEE Communications Surveys & Tutorials, 2017.

[5] H. Zhang und J. Yan, "Performance of SDN routing in comparison with legacy routing protocols," In Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2015.

[6] Y. Jinyao, Z. Hailong, S. Qianjun, L. Bo und G. Xiao, "HiQoS An SDN-based multipath QoS solution ," China Communications, Bd. 12, Nr. 5, p. 123–133, 2015.

[7] J. L. Valenzuela, A. Monleon, I. San Esteban, M. Portoles und O. Sallent, "A hierarchical token bucket algorithm to enhance QoS," in IEEE 802.11, In Vehicular Technology Conference, VTC2004-Fall. 2004 IEEE 60th (Vol. 4, pp. 2659-2662), 2004.

[8] Project Floodlight, "Floodlight OpenFlow Controller," [Online] Available: http://www.projectfloodlight.org/floodlight/.

[9] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat und G. Varghese, "P4 Programming protocol-independent packet processors," ACM SIGCOMM Computer Communication Review, Bd. 44, Nr. 3, p. 87–95, 2014.

[10] Hewlett Packard, "HP Switch Software OpenFlow Administrator's Guide K/KA/WB 15.1," [Online] Available: http://h20628.www2.hp.com/km-ext/kmcsdirect/emr_na-c03991489-1.pdf.

[11] Brocade, "Brocade NetIron Software Defined Networking (SDN) Configuration Guide," [Online] Available: http://www.brocade.com/cotent/html/en/configuration-guide/NI_05800a_SDN/GUID-4C86703D-AF09-43B0-8DCA-8402D65624B0.html.

[12] Juniper Networks, "OpenFlow Support on Juniper Networks Devices," [Online] Available: https://www.juniper.net/documentation/en_US/release-independent/junos/topics/reference/general/junos-sdn-openflow-supported-platforms.html.

[13] Pica8, "PicOS Open vSwitch Configuration Guide," [Online] Available: http://www.pica8.com/wp-content/uploads/2015/09/v2.9/html/ovs-configuration-guide/.

[14] D. Wermser und O. Gebauer, "NaaS as Business Concept and SDN as Technology – How do They Interrelate," 20. ITG-Fachtagung, 2015.