# Network Configuration with Quality of Service Abstractions for SDN and Legacy Networks

Christian Sieber*, Andreas Blenk*, David Hock†, Marc Scheib†, Thomas Höhn†, Stefan Köhler†, Wolfgang Kellerer*
*Chair of Communication Networks, Department of Electrical and Computer Engineering,
Technische Universität München, Germany
{c.sieber, andreas.blenk, wolfgang.kellerer}@tum.de
†Infosim GmbH & Co. KG, Würzburg, Germany
{hock, scheib, thomas.hoehn, koehler}@infosim.net

## I. Introduction and problem description

In this paper, we demonstrate an implementation of a Network Services Abstraction Layer (NSAL) on top of the network control and management plane. Furthermore, we introduce a unified data model for both Software Defined Networking (SDN) and legacy devices that allows managing and configuring both networks in a unified way in order to achieve Quality of Service (QoS) for time-critical tasks (e.g. VoIP). Due to the unified data model, network operators are able to manage their network through one interface. We demonstrate a use case by implementing a VoIP scheduling application on top of the NSAL and evaluate VoIP call quality in a distributed heterogeneous network.

Traditionally, network devices are managed by a logically centralized entity (i.e. the Network Management System (NMS)) and forwarding decisions are made on the devices by distributed control algorithms. In contrast, SDN introduces a split between the control and the forwarding by moving the control logic away from the device into a logically centralized controller. OpenFlow, as a SDN protocol for the communication between controller and forwarding device, introduces forwarding abstractions based on flow matching rules. By pushing the rules to the devices, the controller is able to control the forwarding of data in the network. OpenFlow is an open protocol and follows the vision of enabling vendor-neutral device control. However, the management of the device (i.e. port up/down, QoS management) is still highly dependent on the device and vendor.

The OFCONF data model on top of the generic management protocol NETCONF is the counterpart of OpenFlow for the management plane. OFCONF introduces abstractions and a common data model for OpenFlow-enabled devices. By this means, a NMS is able to manage OpenFlow-enabled devices independent of the vendor of the device. As of writing, OFCONF is still new and the data models limited in its scope. Although protocols such as OFCONF are expected to standardize the way to configure the QoS capabilities of network elements, the available capabilities for QoS remain switch-dependent, i.e., vendor-dependent. This means, that among multiple switches, a large variety of QoS mechanisms, such as schedulers, shapers, and matchers exist. Due to this variety of mechanisms, a unified data model is not available.

Furthermore, there is still limited understanding of how control and management plane should interact. In OpenFlow for example, configuration tasks requiring switch resources (e.g. queues and scheduler) have to be performed by both management and control plane. The control plane steers packets into the queues, while the management plane has the knowledge about the queue features and can configure the scheduling in order to implement QoS in the network. Due to the lack of a unified management and control plane for QoS, the configuration may thus be highly complex, in particular for networks consisting of vendor- and device-varying network hardware. For configuring the individual network elements, networks operators have to use vendor-dependent configuration and management software.

## II. Methodology & Background

We follow the terminology proposed in [1], which defines four planes. A forwarding plane, an operational plane, a management plane and an application plane. We define three types of switches in Table I. The table is sorted by the level of abstraction, where SDN offers the most abstractions and legacy hardware the least. SDN, as combination of OpenFlow for configuration of the forwarding plane and OFCONF for the configuration of QoS, allows for device and vendor-independent configuration of the device. Hybrid switches are in between SDN and legacy switches and offer forwarding abstraction and configuration through OpenFlow. However, there are no device and vendor-independent abstractions for the QoS configuration. Legacy switches are not configured based on strict forwarding or QoS abstractions. Each vendor implements device specific interfaces and abstractions.

TABLE I. Switch types summary

| Type | Description |
|---|---|
| SDN (OpenFlow, OFCONF) | Well-defined interface to forwarding and QoS configuration. E.g. OpenFlow-enabled switch with OFCONF support. |
| Hybrid (OpenFlow, Legacy QoS) | Forwarding abstraction through OpenFlow, proprietary QoS abstractions and configuration. |
| Legacy | Proprietary forwarding and management abstractions and configuration. |

### A. Per-switch Quality of Service Abstractions / Models

Figure 1 shows a example of a simplified QoS configuration model for a hybrid switch. Note that each interface has to be modeled as unidirectional to account for configuration differences depending on the packet flow direction. E.g. queues for egress ports are common and essential for QoS options,

queues for ingress ports are less common in the switches available in our experimental set-up. Also note that each switch's interfaces have to be modeled individually as features can differ between interfaces. E.g. one of the switches in the experimental set-up supports no queues on some of the interfaces. We describe and store the switch QoS models as YANG models in a database indexed by vendor and switch model. The term *Backplane* here describes a placeholder for the routing decision made by the control plane.

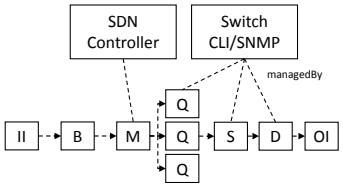


Fig. 1. Simplified QoS model for a hybrid switch. II: Input Interface, B: Backplane, M: Matcher, Q: Queue, S: Scheduler, D: Dropper, OI: Output

Depending on their type, components have one or more input and outputs. For example, a matcher has only one input, but multiple outputs. Furthermore, each switch component has an individual abstraction/data model. For example the matcher stores a list of possible match fields and supports the function *matchTo* which associates one or more match values with a specific output (i.e. match to a queue).

## III. Experimental Setup

The experimental set-up consists of physical and virtual switches of different vendors and covers all three types presented in Table I. As hardware-based hybrid switch with OpenFlow and legacy QoS configuration, we deploy a NEC PF5240 Programmable Flow switch. As legacy switch, we use one Cisco Catalyst 4503-E Switch and one Cisco Catalyst 3650. To the best of our knowledge, there are currently no OFCONF-enabled hardware switches available on the market. Therefor, the SDN switch type is represented by OpenvSwitch software-switches with a custom OFCONF implementation. As NMS, we use StableNet[2] developed by infosim which is extended in northbound direction for communicating with the NSAL and in southbound direction for OFCONF configuration.

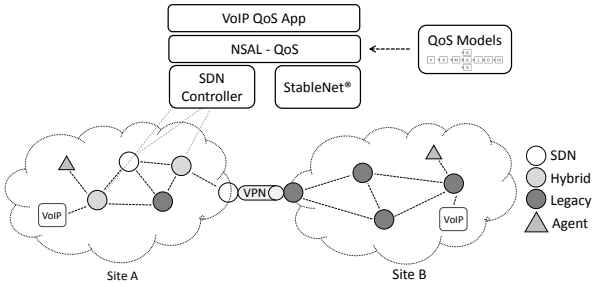


Fig. 2. Example topology with two sites connected through a VPN.

Figure 2 gives an example topology from the demonstration set-up. Two sites, Site A located in Munich and Site B located in Wuerzburg, are connected via a fixed-bandwidth VPN connection. Site A uses a mix of SDN, legacy and hybrid switches to connect multiple VoIP endpoints to the VPN gateway. The VPN tunnel endpoint is connected to a SDN switch. At Site B, there are multiple legacy switches connected to each other, to the VPN gateway and to the VoIP endpoints. At both sites, StableNet measurement agents are deployed to measure the end-to-end VoIP call quality by active probing.

The control plane for the SDN switches is implemented with OpenDaylight as SDN controller at Site A. The management plane for both sites is implemented through StableNet located at Site B. On top of the control and management plane is our Network Services Abstraction Layer for QoS as an independent implementation located at Site A. On top of the NSAL for QoS runs the VoIP QoS application, which uses the NSAL northbound API to implement the required changes in the network for VoIP calls. The VoIP QoS application also acts as a call manager for the VoIP system. Figure 3 shows the topology as discovered by the NMS StableNet and VoIP call statistics from the call manager and the measurement agents.

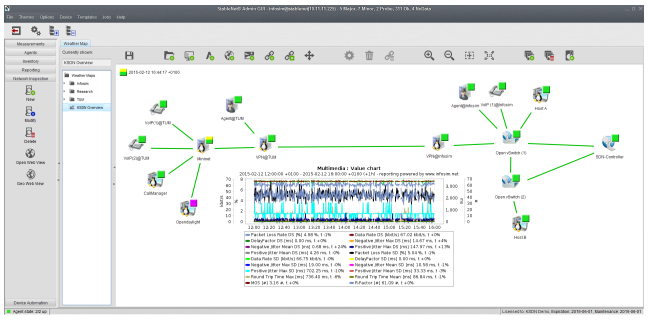

Fig. 3. StableNet discovered topology and measured VoIP call quality.

## IV. Demo Presentation

The demo presentation illustrates how Quality of Service management in heterogeneous environments can profit from a unifying Network Service Abstraction Layer in combination with per-device QoS abstractions. By taking the example of VoIP, we show how a call manager application on top of the NSAL can ensure VoIP call quality in a network with multi-vendor SDN and legacy devices. A GUI shows three different views of the network. The view of the control plane, the view of the management plane and the view of the NSAL. Each possible path between two calling VoIP endpoints is represented as a chain of switch components. A simple greedy algorithm is evaluating the possible paths and chooses a suitable path based on the options and available resources. Through modules for the different involved components, the abstract decisions are implemented in the network. An estimation of the call quality for each call and the chain of components for each call is visualized.

## V. Acknowledgment

This work has been partially funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) under the grant numbers KF3157502LF3 and KF3157602LF3 as part of the ZIM program.

## References

[1] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426 (Informational), Internet Engineering Task Force, Jan. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7426.txt

[2] infosim, "Stablenet," https://www.infosim.net, 2015.