Westfälische
Wilhelms-Universität
Münster

Software-Defined Networking an der Universität Münster

Tim Humernbrum
Institut für Informatik
Arbeitsgruppe Prof. Gorlatch

*Kolloquium Erneuerung des Kommunikationssystems der WWU,
4. November 2016*

1. Quality of Service with SDN

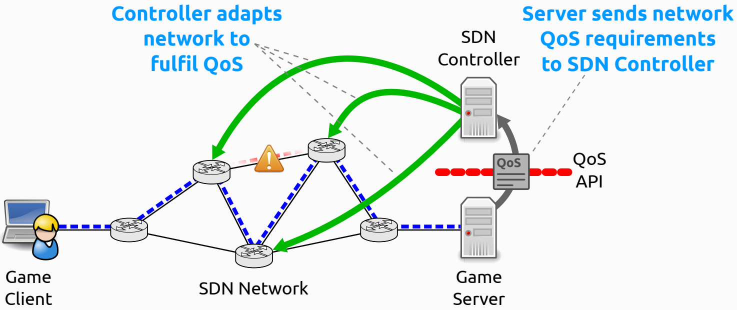
Problems with QoS in Traditional Networks

- Network components (routers, switches) are closed systems
- Control logic managing the forwarding of data packets is integrated in the components
- Only limited possibilities for influencing packet forwarding decisions
- No possibilities to dynamically react to problems like congestion of specific routes
- Current *Quality of Service (QoS)* standards (e. g. IntServ, DiffServ) are static and not widely supported

→ Nearly all applications use the network on a best-effort basis

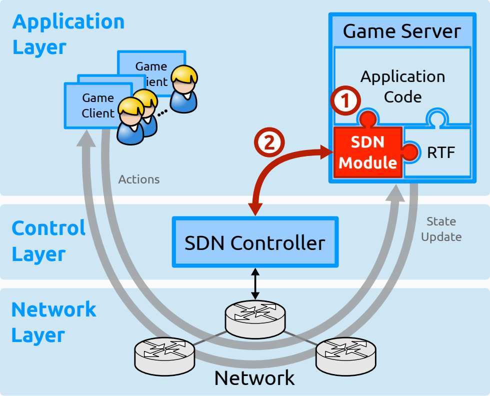
Exploiting SDN for Improving QoS

- With *Software-Defined Networking (SDN)*, applications can dynamically reconfigure the network for their purposes
- **Our goal:** Provide QoS for real-time applications using SDN
- **Our work:** Novel API and its implementation for QoS specification and monitoring
- **Central research challenge:**
  Translation: application QoE → network QoS



QoS API Basic Architecture

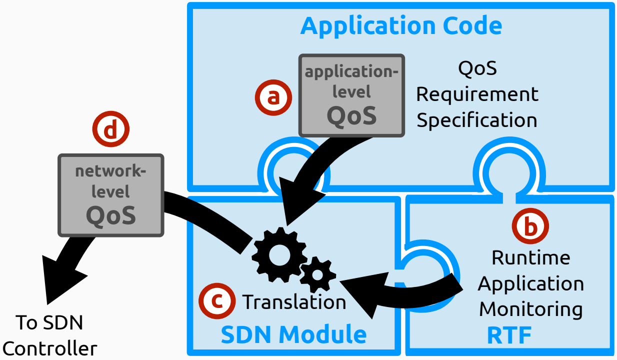
- QoS requirements are specified by the developer for a particular flow using the application-level API ①
- The QoS requirements are automatically reported to the SDN controller by the SDN Module via the network-level API ②



Application-Level Metrics

- Problem: the SDN controller can only understand and manage QoS requirements which are based on network-level metrics, e. g., throughput, packet loss, etc.
- These are too low-level from the developers point of view and not directly related to application-level metrics
- SDN Module provides application-level metrics for the specification of QoS requirements to the developer, e. g., response time
- Automatic translation into network-level metrics at runtime
- Design goal: transparent for the developer, no different usage of application- and network-level metrics in the SDN Module

Basic Translation Concept



Once an application-level requirement has been specified, steps b) – d) are continuously repeated and the translated QoS requirement is adapted if the runtime monitoring data changes

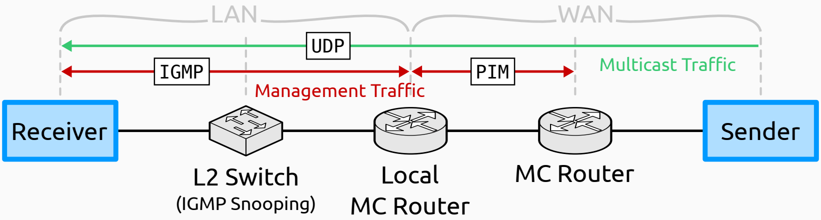
2. Multicast with SDN

Multicast (MC)

Traditional networks: IP multicast, key properties:

- Receiver initiated: receivers subscribe at local MC router (IGMPv3/MLDv2)
- No membership control possible for the sender
- Every host can send to any MC group without joining it
- Distributed MC routing (e. g., using PIM)

→ **Not desirable** in many application use-cases



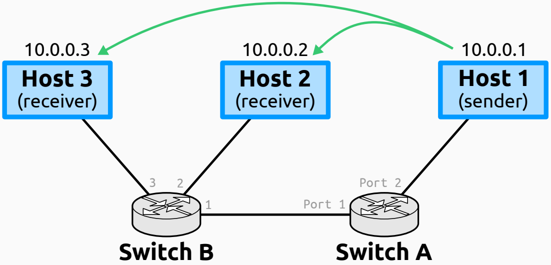
Multicast (MC)

Our approach: multicast exploiting the centralized SDN architecture

- Sender initiated: no action required from the receivers
- Full membership control for the initiator
- Only the initiator is able to send data to the MC group
- Centralized MC routing by the SDN controller

$\rightarrow$ Alternative to IP multicast, not a replacement

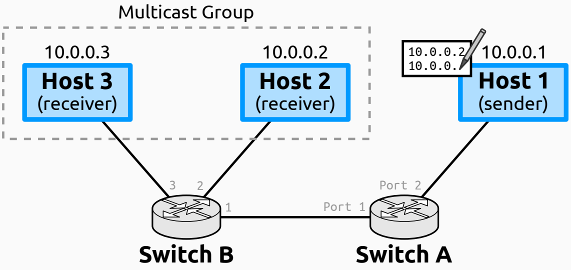
SDN-based Multicast: Example Scenario



Example scenario: host 1 sends data to hosts 2 and 3 using multicast

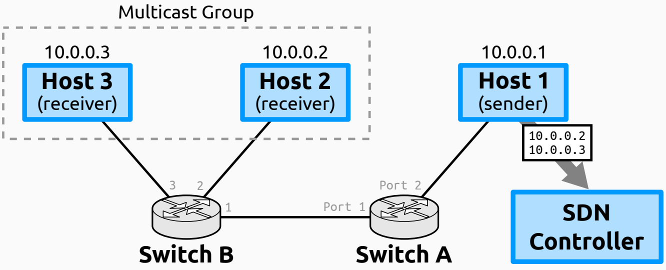
The multicast is initialized in 4 steps

SDN-based Multicast: Step 1



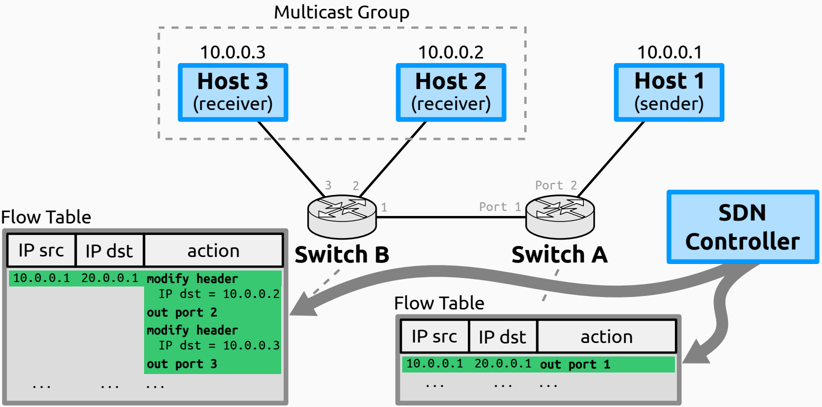
Step1: host 1 specifies a multicast group with IP adresses of 2 and 3

SDN-based Multicast: Step 2



Step 2: the group specification is sent to the SDN controller

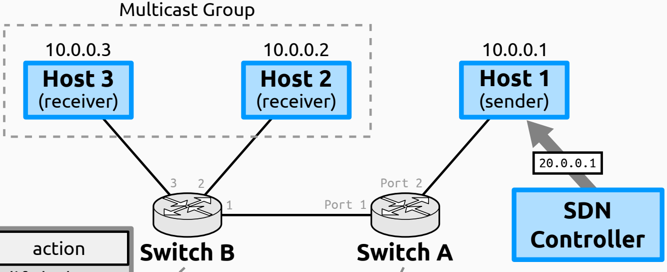
SDN-based Multicast: Step 3



Step 3: the SDN controller installs the multicast in the network

New rules are added to the flow tables of switches A and B

SDN-based Multicast: Step 4



Flow Table

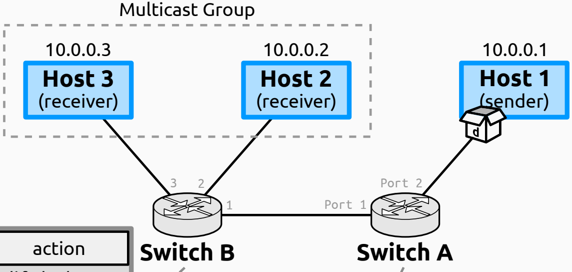
IP src	IP dst	action
10.0.0.1	20.0.0.1	**modify header** IP dst = 10.0.0.2 **out port 2** **modify header** IP dst = 10.0.0.3 **out port 3**
...	...	...

Flow Table

IP src	IP dst	action
10.0.0.1	20.0.0.1	**out port 1**
...	...	...

Step 4: the multicast address is returned by the SDN controller to host 1

SDN-based Multicast: Sending Data



Flow Table

IP src	IP dst	action
10.0.0.1	20.0.0.1	**modify header** IP dst = 10.0.0.2 **out port 2** **modify header** IP dst = 10.0.0.3 **out port 3**
...	...	...

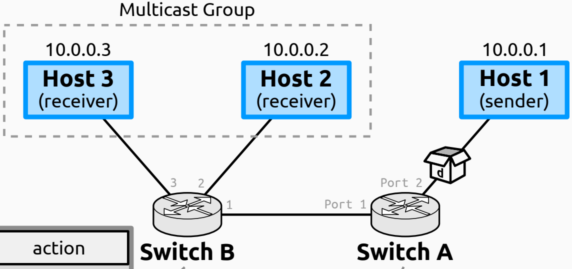
Flow Table

IP src	IP dst	action
10.0.0.1	20.0.0.1	**out port 1**
...	...	...

Host 1 sends a multicast packet to the multicast address

`(d = 20.0.0.1)`

SDN-based Multicast: Sending Data



Flow Table

IP src	IP dst	action
`10.0.0.1`	`20.0.0.1`	**`modify header`** `IP dst = 10.0.0.2` **`out port 2`** **`modify header`** `IP dst = 10.0.0.3` **`out port 3`**
...	...	...

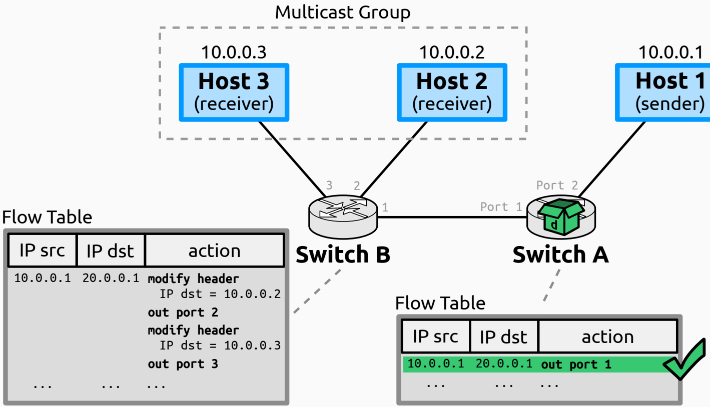
Flow Table

IP src	IP dst	action
`10.0.0.1`	`20.0.0.1`	**`out port 1`**
...	...	...

Host 1 sends a multicast packet to the multicast address

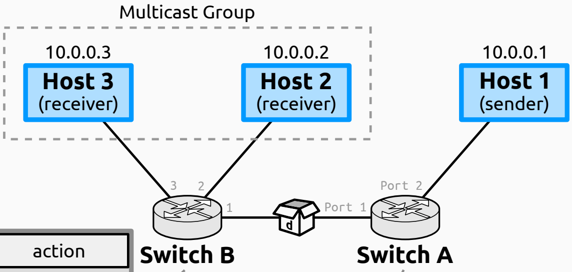
(`d = 20.0.0.1`)

SDN-based Multicast: Sending Data



The packet matches the new rule in switch A and is forwarded to B

SDN-based Multicast: Sending Data



Flow Table (Switch B)

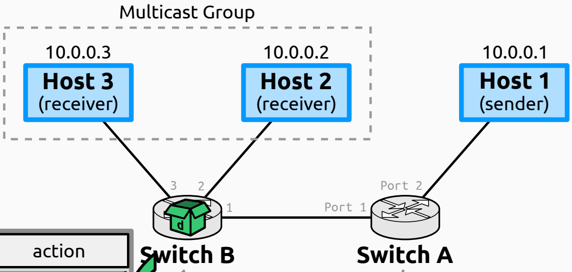
IP src	IP dst	action
10.0.0.1	20.0.0.1	**modify header** IP dst = 10.0.0.2 **out port 2** **modify header** IP dst = 10.0.0.3 **out port 3**
...	...	...

Flow Table (Switch A)

IP src	IP dst	action
10.0.0.1	20.0.0.1	**out port 1**
...	...	...

The packet matches the new rule in switch A and is forwarded to B

SDN-based Multicast: Sending Data



Flow Table (Switch B)

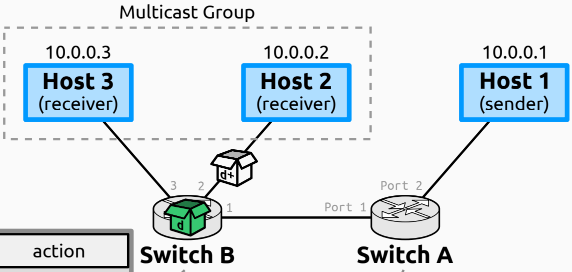
IP src	IP dst	action
10.0.0.1	20.0.0.1	modify header IP dst = 10.0.0.2 out port 2
		modify header IP dst = 10.0.0.3 out port 3
...	...	...

Flow Table (Switch A)

IP src	IP dst	action
10.0.0.1	20.0.0.1	out port 1
...	...	...

In switch B, the multicast paket is transformed into a unicast paket: the multicast address $\mathtt{d}$ is replaced by the unicast address $\mathtt{d}^+$ of host 2

SDN-based Multicast: Sending Data



Flow Table (Switch B)

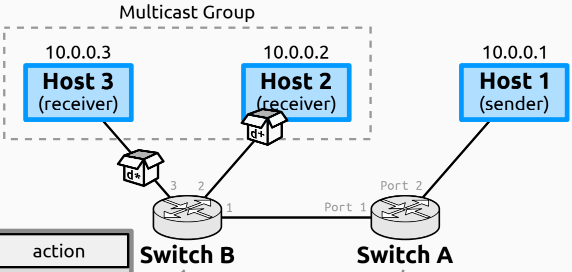
IP src	IP dst	action
10.0.0.1	20.0.0.1	**modify header** IP dst = 10.0.0.2 **out port 2**
		modify header IP dst = 10.0.0.3 **out port 3**
...	...	...

Flow Table (Switch A)

IP src	IP dst	action
10.0.0.1	20.0.0.1	**out port 1**
...	...	...

In switch B, the multicast paket is transformed into a unicast paket: the multicast address `d` is replaced by the unicast address `d*` of host 3

SDN-based Multicast: Sending Data



Flow Table (Switch B)

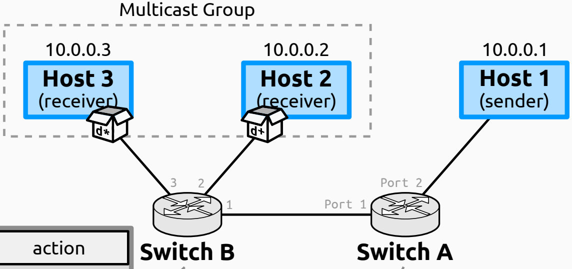
IP src	IP dst	action
10.0.0.1	20.0.0.1	**modify header** IP dst = 10.0.0.2 **out port 2** **modify header** IP dst = 10.0.0.3 **out port 3**
...	...	...

Flow Table (Switch A)

IP src	IP dst	action
10.0.0.1	20.0.0.1	**out port 1**
...	...	...

The modified packets are forwarded to hosts 2 and 3, respectively

SDN-based Multicast: Sending Data



Flow Table (Switch B)

IP src	IP dst	action
10.0.0.1	20.0.0.1	**modify header** IP dst = 10.0.0.2 **out port 2** **modify header** IP dst = 10.0.0.3 **out port 3**
…	…	…

Flow Table (Switch A)

IP src	IP dst	action
10.0.0.1	20.0.0.1	**out port 1**
…	…	…

The modified packets are forwarded to hosts 2 and 3, respectively

Advantages of SDN-based Multicast

Sender initiated

- Sender specifies the MC group and initiates the MC
- Receivers can be added and removed dynamically
- Minimized management overhead in the network: no IGMP needed

Receiver transparent

- MC packets are transformed into unicast packets before their delivery
- Minimized frame filtering in receivers NIC and IP stack
- Receivers cannot send to MC group without sender

Also works with IPv6 (requires OpenFlow 1.2 support)

Multicast Trees

Potential problem of SDN-based MC (also of IP MC):

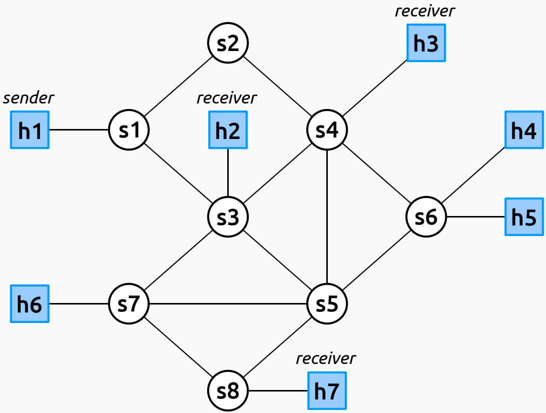
- MC forwarding rules are stored in the switches
- Flow table size is a potential bottleneck for MC scalability

$\rightarrow$ limited number and/or size of MC groups

Central research challenges:

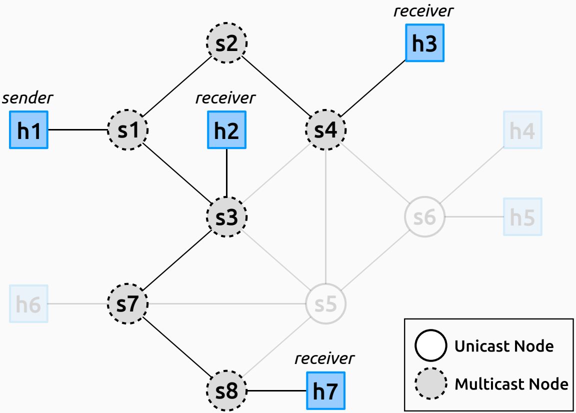
- Reduce the number of switches for realizing MC
- Save flow table space by reusing unicast entries

Multicast Trees



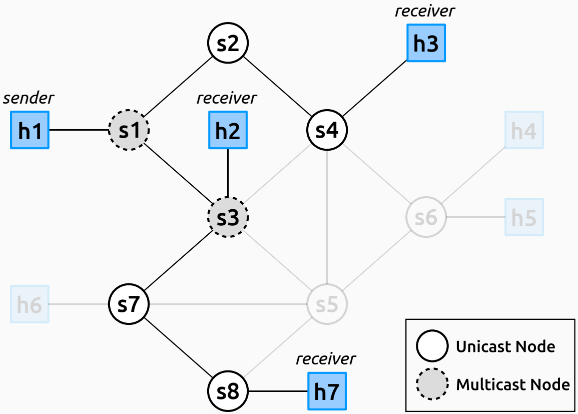
Traditional MC routing: find a path to all receivers of a MC group by using *Shortest-Path Trees (SPT)*

Multicast Trees



Traditional MC routing: find a path to all receivers of a MC group by using *Shortest-Path Trees (SPT)*

Multicast Trees



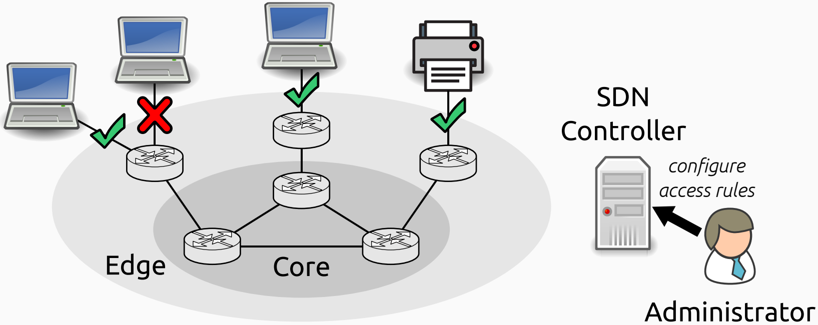
Applying our so-called *Branch-aware Modification (BAM)* to an SPT allows for reusing unicast entries

3. SDN-Pilotprojekt zur Netzsicherheit

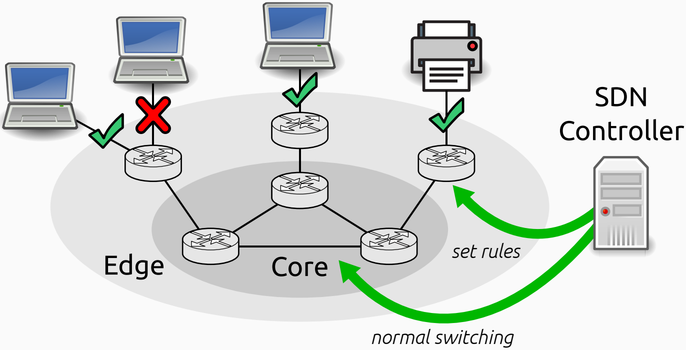
Zugangsschutz zum LAN mit SDN

- Kooperation zwischen dem Institut für Informatik und dem ZIV
- Bachelorarbeit zum Thema Zugangsschutz mit SDN als Alternative zu IEEE 802.1X
- Motivation: Erfahrungsaufbau mit SDN
- Ziele des Projekts:
  - Endgeräte anhand ihrer MAC-Adresse autorisieren
  - Einfache, zentrale Konfiguration
  - Gruppierung von Switches/Ports zu Zugangsbereichen
  - Filtern bestimmter Diensten
  - Keine zusätzliche Software für die Endgeräte
- Umsezung als Modul für den *HPE VAN SDN Controller*
- Fortsetzung des Projekts in Abschlussarbeiten und Seminaren

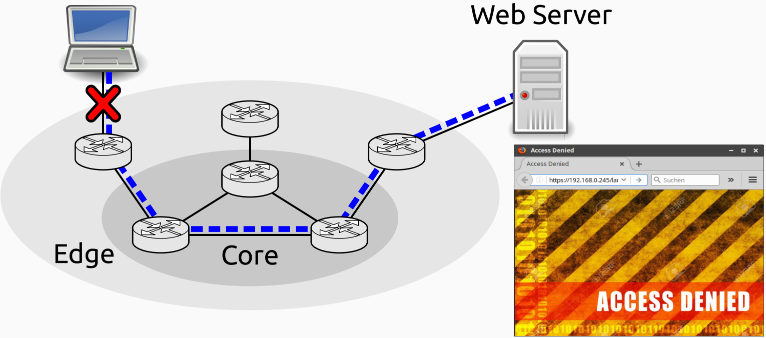
Konzept des Zugangsschutzes mit SDN



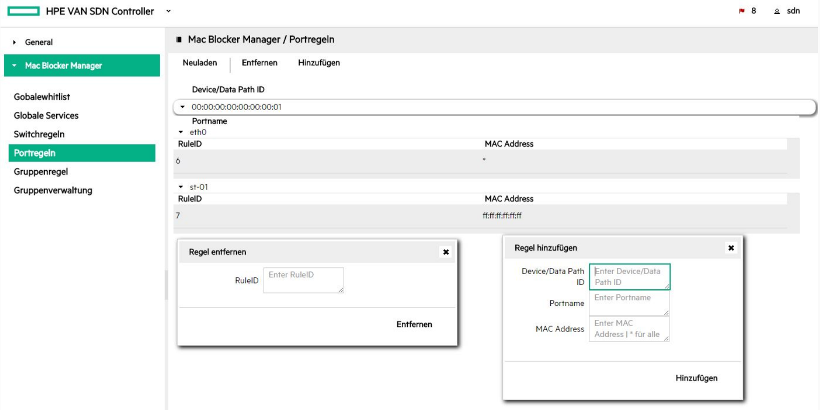
Konzept des Zugangsschutzes mit SDN



Konzept des Zugangsschutzes mit SDN



Konfiguration der Zugangsschutz-App



Web-Portal zur Konfiguration des Zugangsschutzes für den Administrator

Vielen Dank!