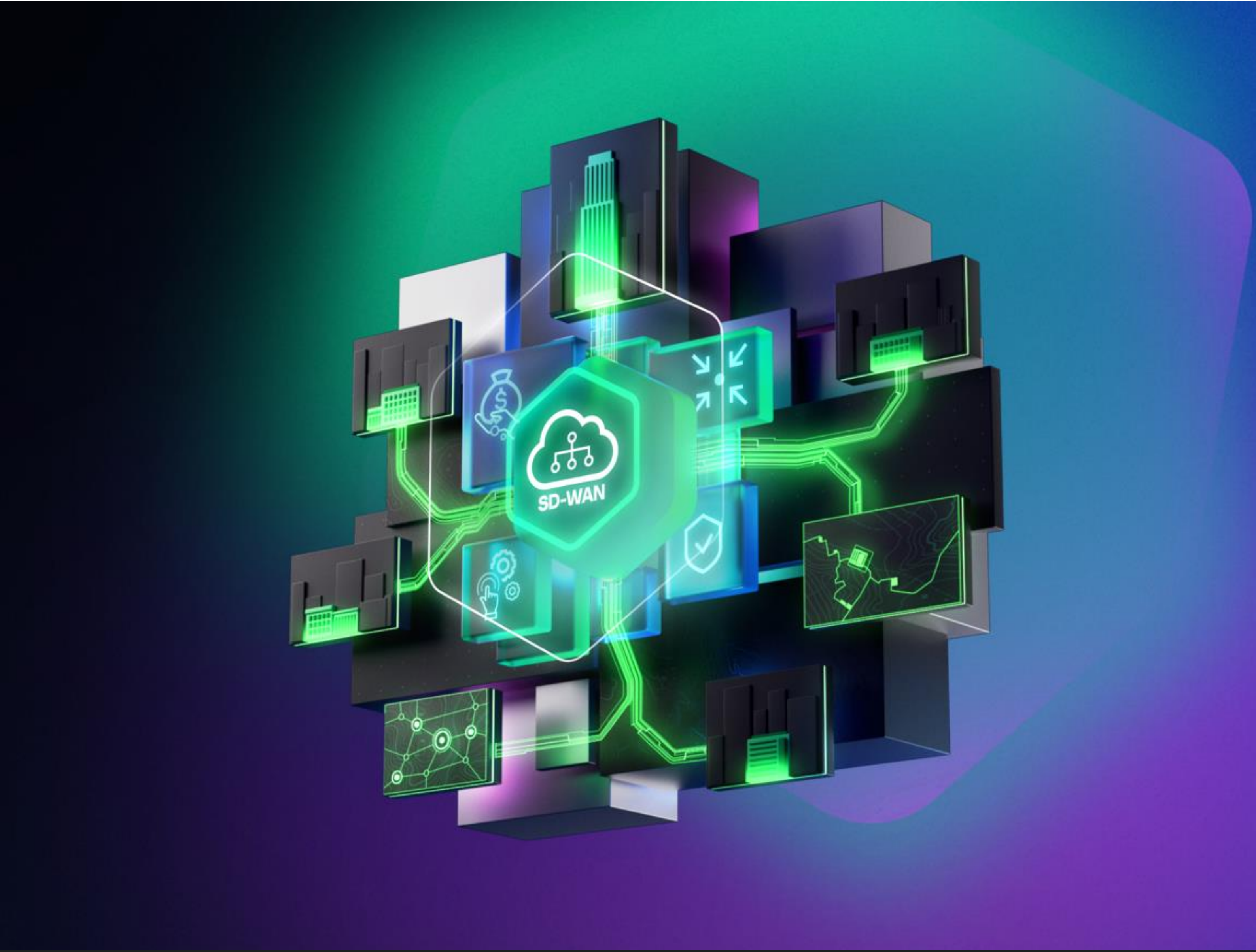


Kaspersky SD-WAN

26.12.2024

Document version: 2.3.1.0

Proof of Concept

Part 1: advanced traffic management, load balancing, application prioritization / SLA, forward error correction, full / partial-mesh topologies

kaspersky bring on the future

Center of Corporate Business Expertise

Change log

Date	Info
05.12.2023	Initial version.
15.12.2023	Updated case descriptions.
31.05.2024	Removed VRRP (moved to Part 1), removed CPE upgrade (moved to CPE maintenance guide), update for release 2.2.1.
19.08.2024	Update after review.
26.12.2024	Update for release 2.3.1.

Contents:

1. Kaspersky SD-WAN 4
1.1. Kaspersky SD-WAN Architecture 5
2. Description of the Kaspersky SD-WAN PoC 6
2.1. Kaspersky SD-WAN PoC topology 7
2.2. PoC IP address plan and resource requirements for SD-WAN components 8
2.3. Network ports used by core system components 10
2.4. SD-WAN containers' external connections diagram 11
2.5. Software versions 12
2.6. Hardware requirements for the Kaspersky SD-WAN solution 12
3. Traffic management 13
3.1. Load balancing in Active / Active link mode 14
3.2. Redundancy with Active/Standby link mode 20
3.3. Packet loss overcome with packet duplication in broadcast mode 25
3.4. Improving network channels reliability through Forward Error Correction 29
3.5. Link quality monitoring (Jitter, Latency, Packet Loss) and traffic management 36
3.6. Traffic prioritization with ACLs 44
3.7. Traffic prioritization with DPI 53
4. SD-WAN Topology Configuration 66
4.1. Creating Full-Mesh topology 67
4.2. Creating Partial-Mesh topology 70
4.3. Creating topologies with transit CPEs 73
Appendix A. PoC Checklist 77

1. Kaspersky SD-WAN

A software-defined wide area network (SD-WAN) is a wide area network that uses software-defined network technology, such as communicating over the Internet using encrypted overlay links for distributed company networks.

A key application of SD-WAN is to allow companies to build higher-performance WANs using lower-cost and commercially available Internet access, enabling them to partially or wholly replace more expensive private WAN connection technologies such as MPLS.

Kaspersky SD-WAN addresses the key shortcomings of the existing traditional WAN networks. The Kaspersky SD-WAN solution is a replacement for traditional networking approach, standard WAN routers, provides predictable, optimized access to business-critical applications, is agnostic to WAN transport technologies, and can use any WAN links.

The Kaspersky SD-WAN solution allows you to build a reliable, geographically distributed, and fast-time scalable corporate network with application-aware efficient routing and simplified centralized management.

Kaspersky SD-WAN combines the following key features:

- Centralized, on-prem or cloud-delivered management, multi-tenancy, and Role Based Access Control (RBAC).
- Template-based Zero Touch Provisioning (ZTP) to speed up the connection of new company sites and remove human error.
- High Availability with prioritization of critical business-applications.
- Load-balancing via multiple WAN links.
- Full mesh and partial mesh topologies.
- Network security functions deployment as Virtualized Network Functions (VNFs) and integration into user traffic chains.
- Intelligent traffic management.

1.1. **Kaspersky SD-WAN Architecture**

Description of the main components of Kaspersky SD-WAN:

- SD-WAN orchestrator provides a graphical management interface, CPE device inventory, configuration templates, transport service policies, and CPE device registration.
- SD-WAN controller manages overlay network, builds transport services, performs link quality monitoring, automatically switches application traffic to backup channels, and performs CPE management via the OpenFlow protocol.
- SD-WAN gateways are CPE devices with an assigned SD-WAN gateway topology role. Terminates overlay links from CPE devices and forms an SDN fabric in the form of an overlay network.
- Kaspersky Edge Service Routers (KESR) / Customer Premise Equipment (CPE) - telecommunications equipment that connects to SD-WAN gateways and other CPE devices using overlay links and forward data traffic.

The architecture of Kaspersky SD-WAN is presented in Figure 1.

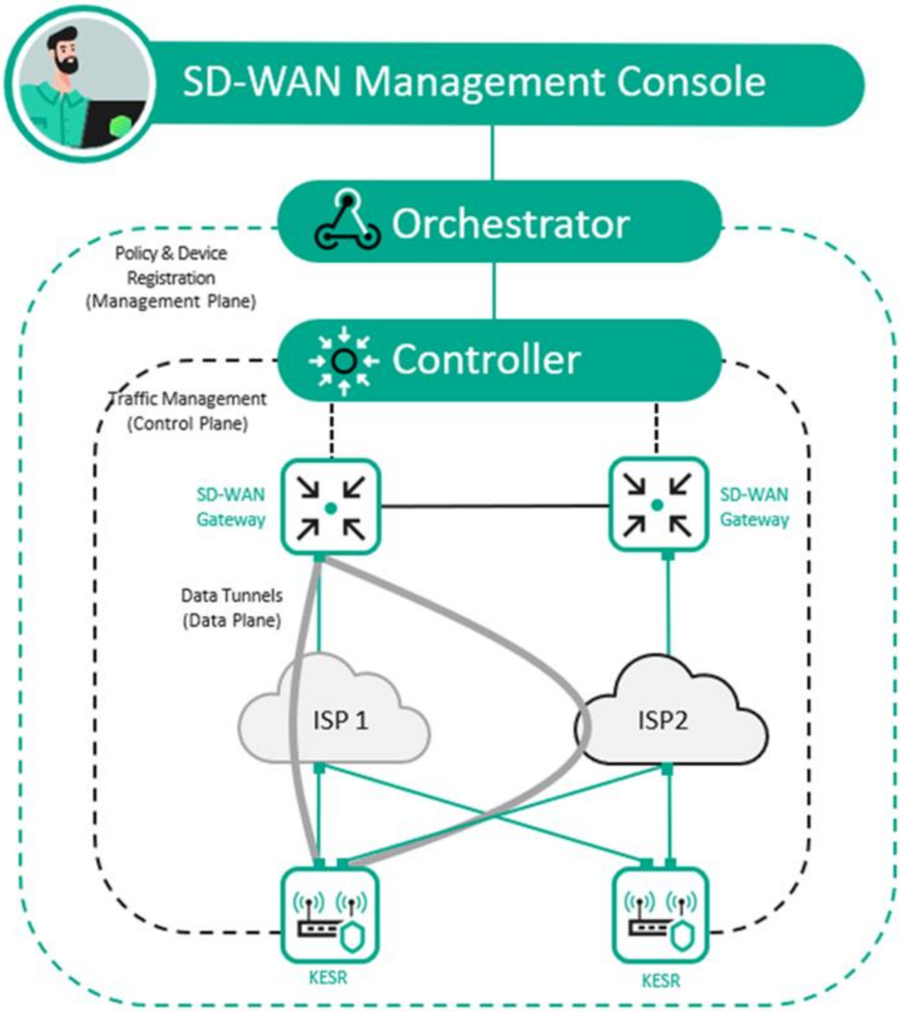


Figure 1 - Kaspersky SD-WAN architecture

2. Description of the Kaspersky SD-WAN PoC

The Kaspersky SD-WAN PoC components are deployed in a VMWare virtualization environment.

The Kaspersky SD-WAN management components such as SD-WAN orchestrator, controller, and monitoring system are deployed as Docker containers on the orc1 virtual host.

Figure 2 shows the topology of the Kaspersky SD-WAN PoC.

PoC topology description:

- The DC site has two network segments, dc-lan1 and oob, that are connected to router R13. The orc1 virtual machine is connected to the oob segment, while the srv1 server is connected to the dc-lan1 segment and hosts the test WWW service.
- There are two routers, R11 and R12, at the DC boundary. Behind them are two SD-WAN gateways: vGW-11 and vGW-12. The internal (lan) interfaces of R13, vGW-11, and vGW-12 are connected to the dc-perim network segment.
- Routers R11 and R12 perform Source Network Address Translation (SNAT) for vGW-11 and vGW-12 and Destination Network Address Translation (DNAT) for the ports specified in Table 1 for connecting CPE devices.
- Router R14 carries out SNAT and acts as the default gateway for Router R13. Additionally, it serves as an Internet gateway for the host orc1. Router R14 provides DNAT for the SD-WAN orchestrator and SD-WAN controller, the ports are specified in Table 1.
- The ISP host emulates the connection to the Internet with ISP1 - ISP8 service providers.
- The vCPE-3 device is an example of a remote site with one CPE device connected to two carriers.
- The vCPE-4 device represents a future scenario in which a remote site is connected using a universal uCPE device that is not currently part of this PoC.
- The vCPE-51 and vCPE-52 gateways are an example of High-Availability scenario. The CPE devices support the VRRP protocol.

2.1. Kaspersky SD-WAN PoC topology

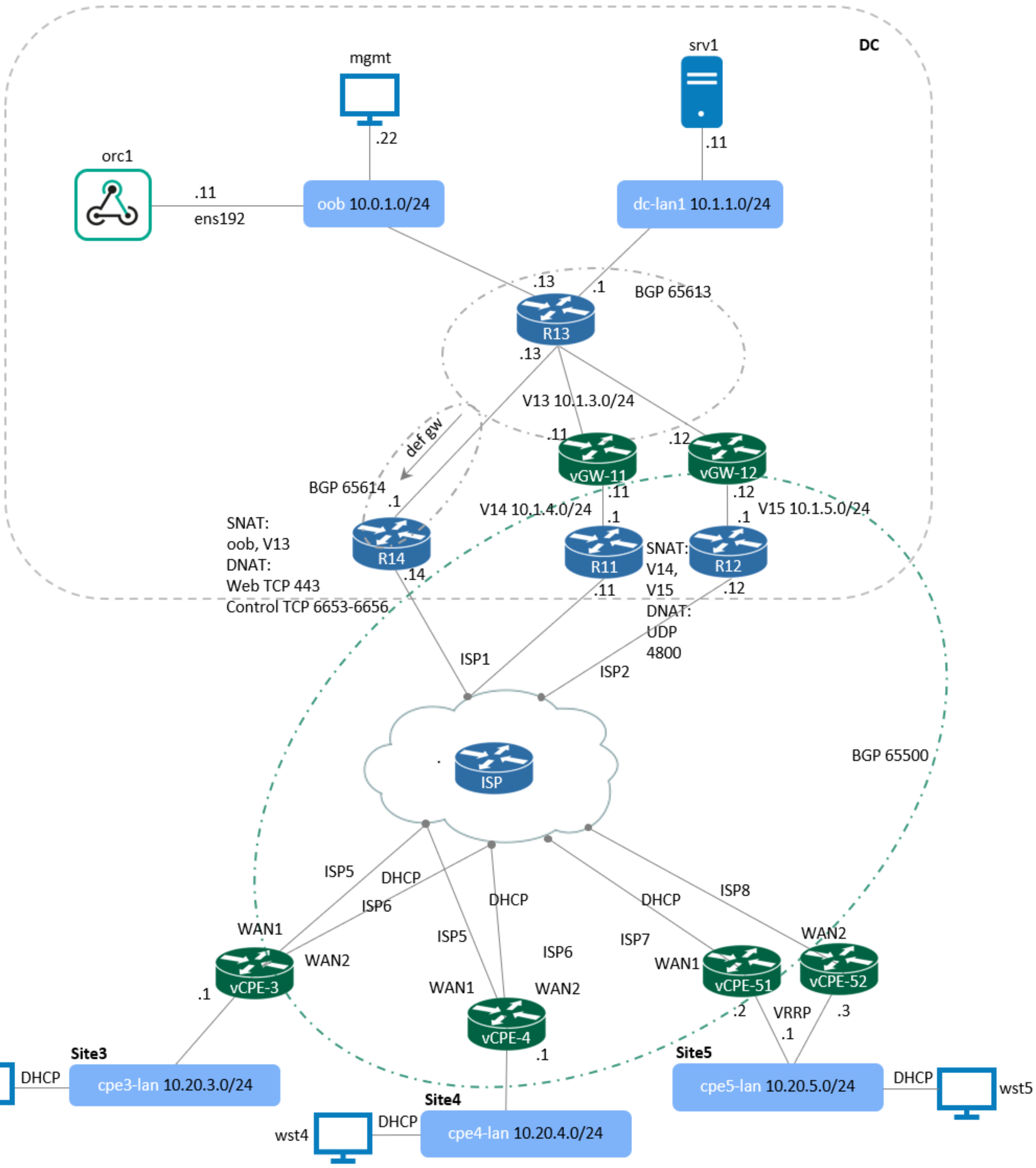


Figure 2 - Kaspersky SD-WAN PoC topology

kaspersky

2.2. **PoC IP address plan and resource requirements for SD-WAN components**

This IP plan corresponds to the scheme in paragraph 2.1. If other addresses are used, it is necessary to change the IP plan and all SD-WAN settings in further steps.

Table 1 - Host parameters used in the PoC

Host	**Operation system**	**IP address**	**Description**	**System requirements**
orc1	Ubuntu 22.04.05 LTS Server	10.0.1.11	Docker containers: www-1, orc-1, redis-1m, mongo-1, vnfm-1, vnfm-proxy-1, ctl-1, zabbix-www-1, zabbix-srv-1, zabbix-proxy-1, zabbix-db-1, syslog-1, mockpnf-1	24 x vCPU, 24 GB RAM
vGW-11	vKESR-M2 image	wan 10.1.4.11 lan 10.1.3.11	SD-WAN gateway	4 x vCPU, 8 GB RAM
vGW-12	vKESR-M2 image	wan 10.1.5.12 lan 10.1.3.12	SD-WAN gateway	4 x vCPU, 8 GB RAM
vCPE-3	vKESR-M1 image	wan DHCP lan 10.20.3.1	CPE	2 x vCPU, 512 Mb RAM
vCPE-4	vKESR-M1 image	wan DHCP lan 10.20.4.1	CPE	2 x vCPU, 512 Mb RAM
vCPE-51	vKESR-M1 image	wan DHCP lan 10.20.5.2 / vIP 10.20.5.1	CPE	2 x vCPU, 512 Mb RAM
vCPE-52	vKESR-M1 image	wan DHCP lan 10.20.5.3 / vIP 10.20.5.1	CPE	2 x vCPU, 512 Mb RAM
R11	CentOS 7	wan 10.50.1.11 lan 10.1.4.1	DC border router	2 x vCPU, 2 GB RAM
R12	CentOS 7	wan 10.50.2.12 lan 10.1.5.1	DC border router	2 x vCPU, 2 GB RAM

Host	Operation system	IP address	Description	System requirements
R13	CentOS 7	dc-perim 10.1.3.13 oob 10.0.1.13 dc-lan1 10.1.1.1	DC core router	2 x vCPU, 2 GB RAM
R14	CentOS 7	wan 10.50.1.14 lan 10.1.3.1	DC border router and NAT	2 x vCPU, 2 GB RAM
ISP	CentOS 7	isp1 10.50.1.1 isp2 10.50.2.1 isp5 10.50.5.1 isp6 10.50.6.1 isp7 10.50.7.1 isp8 10.50.8.1	Emulation of ISP1-ISP8	2 x vCPU, 2 GB RAM
srv1	CentOS 7	10.1.1.11	WWW / DC server	2 x vCPU, 4 GB RAM
wst3	CentOS 7	DHCP 10.20.3.0/24	Site3 workstation	2 x vCPU, 4 GB RAM
wst4	CentOS 7	DHCP 10.20.4.0/24	Site4 workstation	2 x vCPU, 4 GB RAM
wst5	CentOS 7	DHCP 10.20.5.0/24	Site5 workstation	2 x vCPU, 4 GB RAM
mgmt	Windows Server 2022	10.0.1.22 10.1.1.22 10.1.3.22 10.50.1.22 10.20.3.22 10.20.4.22 10.20.5.22	Management workstation	6 x vCPU, 6 GB RAM

2.3. **Network ports used by core system components**

Table 2 – Network ports used by SD-WAN gateways and CPE devices to communicate with the core components of the solution and to access the orchestrator web interface for administration.

Component	**Ports**	**Description**
SD-WAN orchestrator	TCP 443 / TLS	Access to the orchestrator web interface. CPE connection to the orchestrator.
SD-WAN controllers	TCP 6653-6656 / TLS	SD-WAN gateways and CPE devices connection to the controller over TLS. CPE device is connected by each WAN interface to a separate port of the controller: • sdwan0 - 6653 • sdwan1 - 6654 • etc.
Zabbix	TCP 85 / TLS TCP 10051 / TLS	Access to the Zabbix web interface. CPE Zabbix agent connections to the monitoring system.
SD-WAN gateways	UDP 4800-4803	Data traffic

kaspersky

2.4. **SD-WAN containers' external connections diagram**

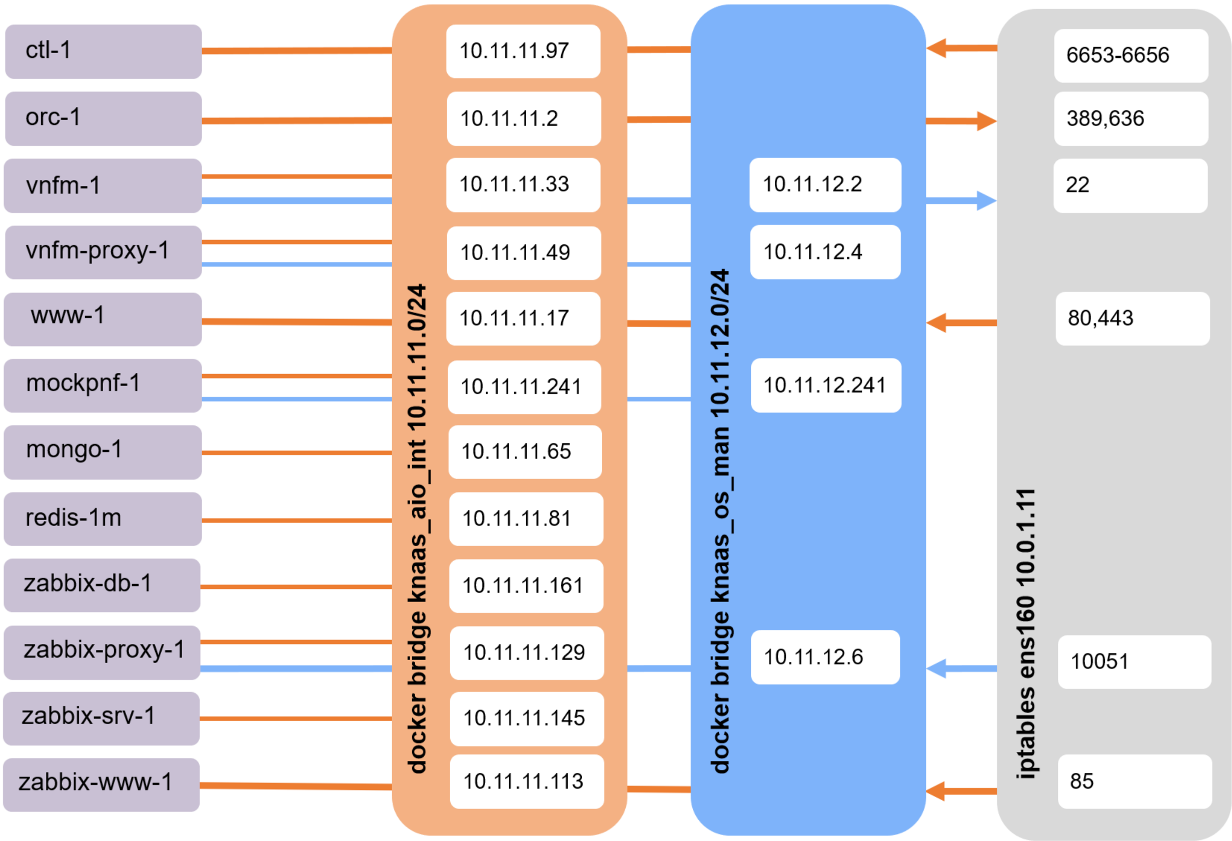


Figure 3 - SD-WAN containers' external connections

SD-WAN containers will be deployed on the orc-1 host. The deployment playbooks create two Docker networks for the containers: **knaas_aio_int** (10.11.11.0/24) and **knaas_os_man** (10.11.12.0/24).

knaas_aio_int is the primary network and is used for communication between containers, as well as for communication with external hosts. **knaas_os_man** is intended for communication between the central components of the solution and CPEs. This network is used for CPE management and monitoring.

In addition, deployment playbooks create iptables rules. Iptables rules are added to the DOCKER_USER chain to allow the following TCP connections:

- Inbound connections to the ctl-1 container on ports 6653-6656 (TLS connections from the CPE devices to the controller).
- Outbound connections from the orc-1 container on ports 389,636 (LDAP/LDAPS connections to the LDAP server).
- Outbound connections from the vnfm-1 container on port 22 (SSH console to CPE from the SD-WAN orchestrator interface).
- Inbound connections to the www-1 container on ports 80 and 443 (HTTPS/TLS connection to the orchestrator web interface and connections from CPEs to orchestrator).
- Inbound connections to the zabbix-proxy-1 container on port 10051 (CPE and VNF monitoring).
- Inbound connections to the zabbix-www-1 container on port 8443 (HTTPS/TLS connection to the Zabbix monitoring system web interface).

2.5. **Software versions**

Table 3 - Versions of Kaspersky SD-WAN software used in this PoC.

SD-WAN component	Version
www	knaas-www:2.24.09.release.65.amd64_en-US_ru-RU
orc	knaas-orc:2.24.09.release.76.amd64_en-US_ru-RU
mongo	mongo:5.0.7.amd64
ctl	knaas-ctl:2.24.09.release.25.amd64_en-US_ru-RU
vnfm	knaas-vnfm:2.24.09.release.15.amd64_en-US_ru-RU
vnfm-proxy	knaas-vnfm-proxy:2.24.09.release.6.amd64_en-US_ru-RU
redis	redis:6.2.7.amd64
zabbix-www	zabbix-web-nginx-mysql:6.0.23.amd64
zabbix-proxy	zabbix-proxy:6.0.23.amd64
zabbix-srv	zabbix-server:6.0.23.amd64
zabbix-db	mariadb-ha:11.1.6.amd64
syslog	syslog-ng:3.30.1.amd64
vCPE	knaas-cpe_2.24.09.release.28
mockpnf	mockpnf: 2.23.09.amd64
orc1 host	Ubuntu 22.04.05 LTS Server
installer	knaas-installer_2.24.09.release.33.amd64_global_en-US_ru-RU

2.6. **Hardware requirements for the Kaspersky SD-WAN solution**

When deploying Kaspersky SD-WAN PoC in an all-in-one deployment scheme with 50 CPE devices, the hardware resources must meet the parameters described in the table below.

Table 4 - Hardware requirements for management of up to 50 CPE devices.

Host	CPU (hyper-threading), cores	RAM, GB	Disk, GB, SSD
orc1	16 cores / 16 vCPU (HT disabled) / 32 vCPU (HT enabled)	32	50 (this value used in the PoC) 684 (recommended)

For more information, please refer to Kaspersky SD-WAN Online Help: https://support.kaspersky.com/help/SD-WAN/2.3/en-US/239105.htm

3. Traffic management

The CPEs are connected via GENEVE tunnels over underlay networks. Links (Tunnels) are unidirectional, so when establishing a connection between two devices, both an inbound link and an outbound link must be created. Links established between CPE devices are combined into a topology.

The set of Links connecting two CPE devices is a segment. Traffic can be distributed over multiple Links on the sender CPE device at the beginning of the segment and forwarded to the receiver CPE device at the end of the segment.

The routes over which traffic can be transmitted within a segment are called transport paths. The following types of transport paths are supported:

- Auto-SPF (Shortest-Path Forwarding). A transport path automatically calculated by the SD-WAN controller. Transport paths of this type cannot be added or removed, and their parameters cannot be changed.
- Manual-TE (Traffic Engineering). A transport path added manually. To add a transport path of this type, you must specify the parameters of the links through which the transport path will pass from the CPE at the beginning of the segment to the CPE at the end of the segment.
- Auto-TE. A transport path automatically calculated by the SD-WAN controller, considering pre-configured constraints. The constraints can be the values of monitoring indicators on links, such as link utilization rate.

Transportation paths have the following parameters:

- Cost. By default, it is the sum of the cost of all links that is included in the transportation path. The ability to manually define the cost of transportation paths is supported.
- Weight.
- Administrative state. This is set manually. If this parameter is set to down, the transport path is not used.
- Operational state. Depends on the presence or absence of the possibility of traffic transfer. If this parameter is set to down, the transport path is not used.

One segment can contain between 2 and 16 transport paths, the best transport path with the lowest value of the cost attribute will be automatically selected for transferring the traffic. If the best transport path is not available for traffic transfer due to technical reasons, another transport path with the best cost value is selected.

For more information, please refer to Kaspersky SD-WAN Online Help: https://support.kaspersky.com/help/SD-WAN/2.3/en-US/250984.htm

kaspersky

3.1. **Load balancing in Active / Active link mode**

Kaspersky SD-WAN provides protection against communication failures between CPE devices using all available network channels simultaneously. The following redundancy modes are supported: Active/Active and Active/Standby.

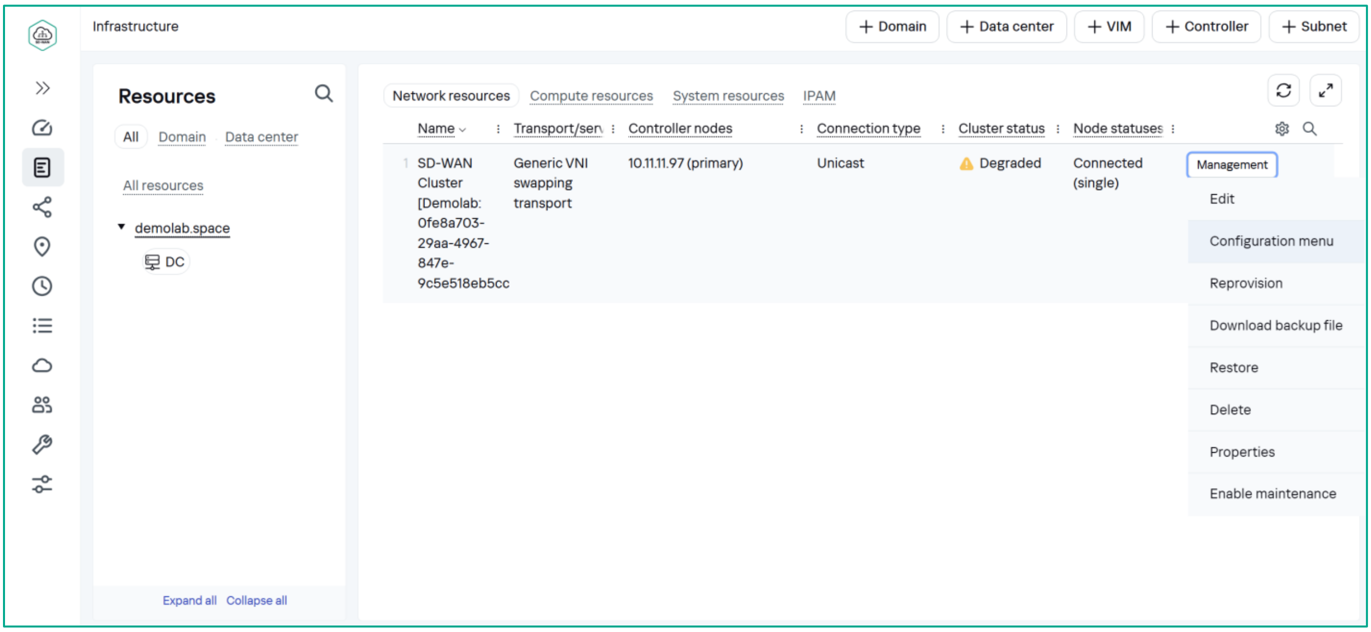
For more information, please refer to Kaspersky SD-WAN Online Help: https://support.kaspersky.com/help/SD-WAN/2.3/en-US/250984.htm

This section demonstrates the load balancing between interfaces on the vCPE-3 device using a pair of WAN interfaces in Active / Active mode. The Cost links parameter is used for load balancing.

Load balancing between the vCPE-3 and vCPE-4 devices for the wst3 and wst4 workstations will be demonstrated using the iperf3 tool. The built-in monitoring system will be used to verify the load balancing operation.

3.1.1. Display SD-WAN fabric segments.

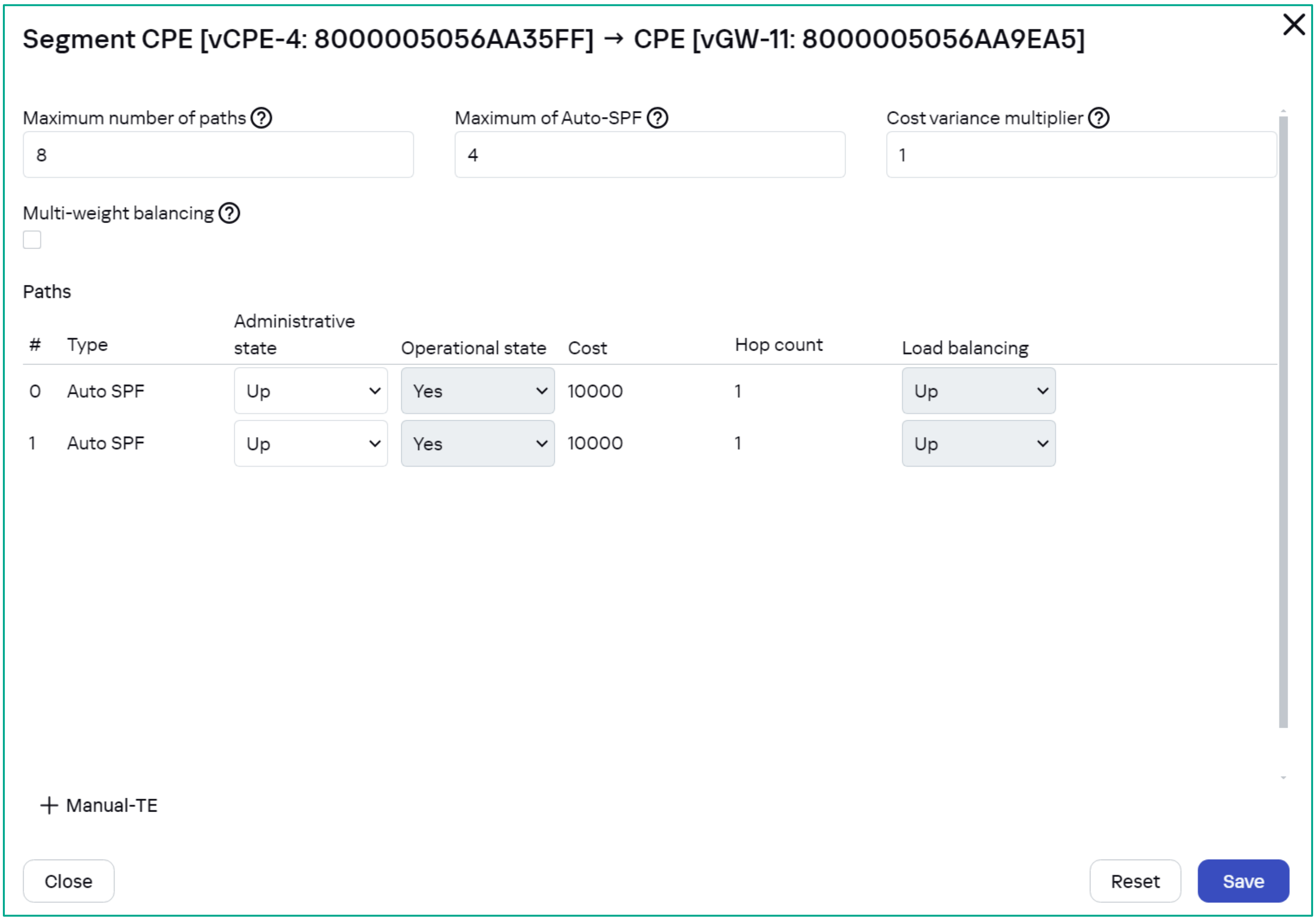
To display a list of all SD-WAN fabric segments, go to **Infrastructure → SD-WAN cluster → Configuration menu → Segments**.



The next screenshot shows an example of a segment between CPEs. Segments are established through CPEs with the Gateway role (vGW-11 and vGW-12). The number of established Auto-SPF links is 2, according to the settings in the CPE template.

Traffic balancing is performed using the OpenFlow select groups.

For more information about segment parameters, click **Management → Edit**.



The controller calculates all possible transport paths in advance, including backup paths, for example, if the actual number of transport paths is greater than the maximum number of Auto-SPF paths set for a particular segment. As soon as a link failure event is detected between CPE devices, the link will be removed from the topology and traffic will be redirected to a backup transport path.

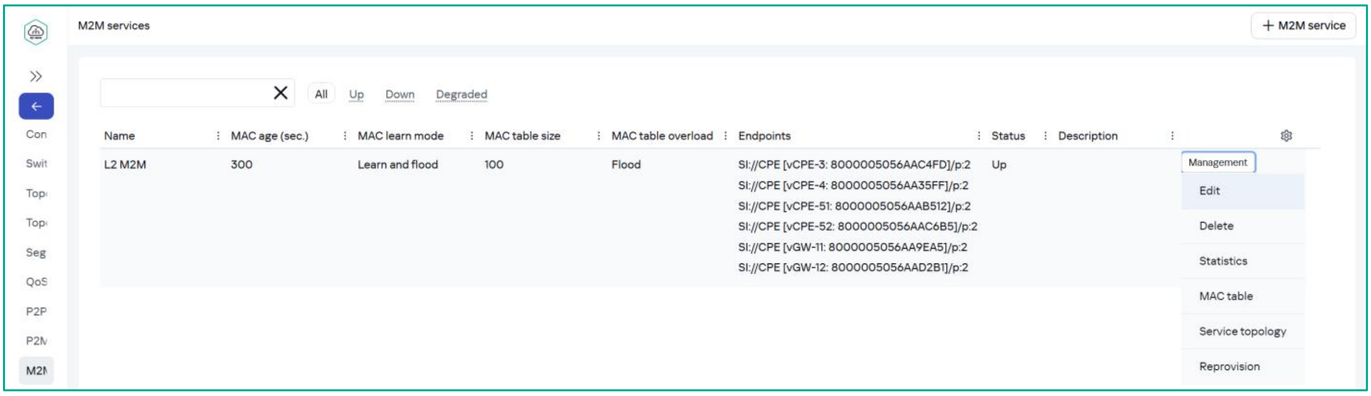
3.1.2. Enable per-packet balancing mode for M2M transport service.

Available balancing modes:

- Per-flow (session) balancing. During transmission, flows are distributed evenly across links.
- Per-packet balancing. Packets are distributed evenly across links during transmission.
- Broadcast. Packets are transmitted simultaneously to all links to eliminate losses.

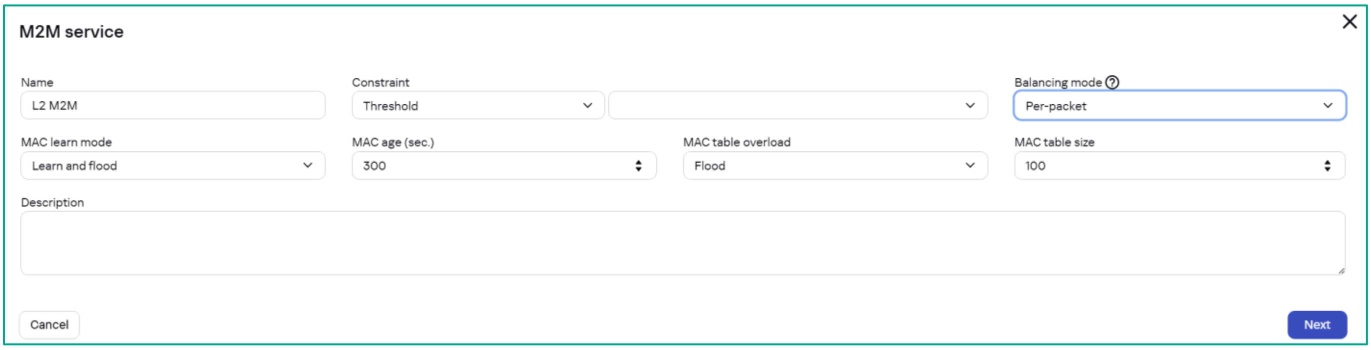
To configure load balance for transport service, go to the transport service settings.

Go to **M2M services**. Select the **L2 M2M** service to edit, then click **Management → Edit**.



This scenario requires **per-packet** balancing mode to be enabled because the scenario uses iperf running on a single port to generate traffic. When per-flow balancing mode is used, only one WAN interface of the CPE device is used.

Select **Balancing mode**: **Per-packet**.

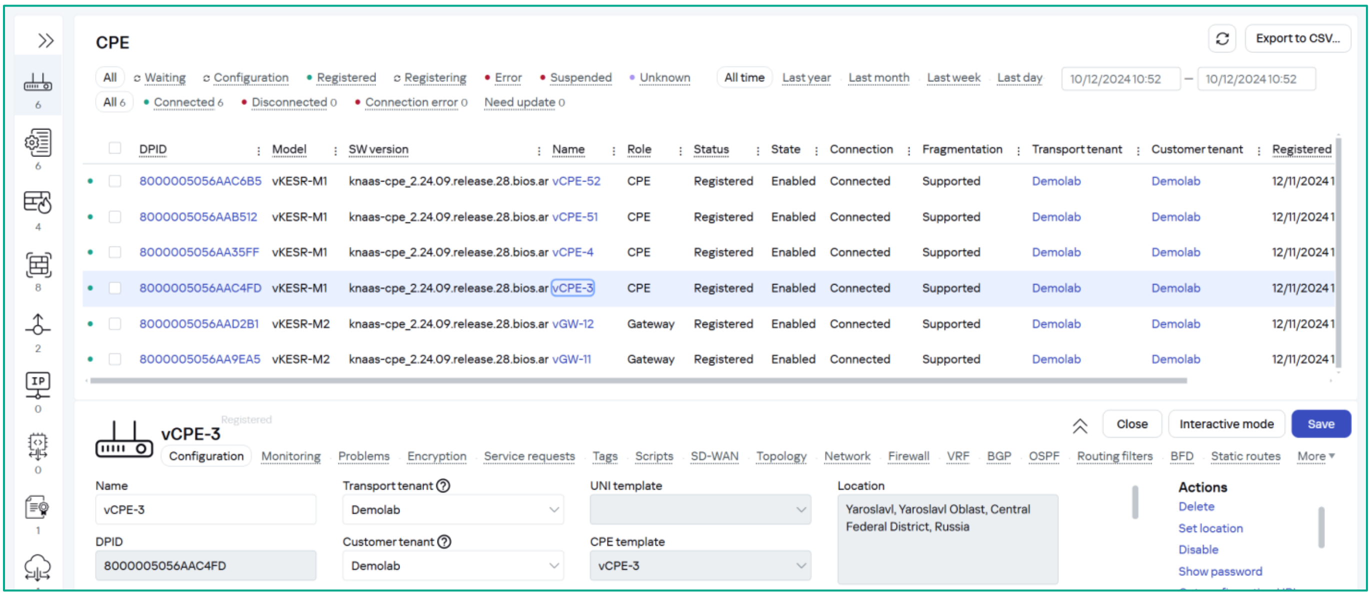


Click **Next**, **Next** then **Save**.

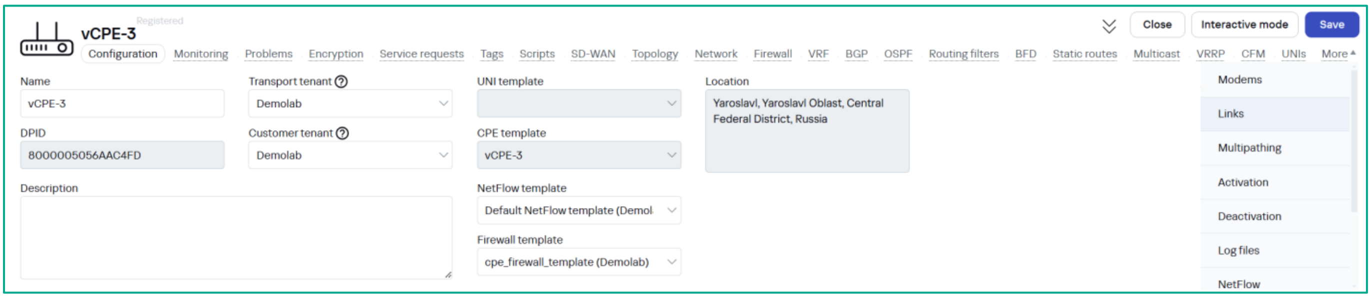
For more information, please refer to Kaspersky SD-WAN Online Help: https://support.kaspersky.com/help/SD-WAN/2.3/en-US/245696.htm

3.1.3. Check vCPE-3 links cost.

Navigate to the **CPE** menu and select **vCPE-3**.

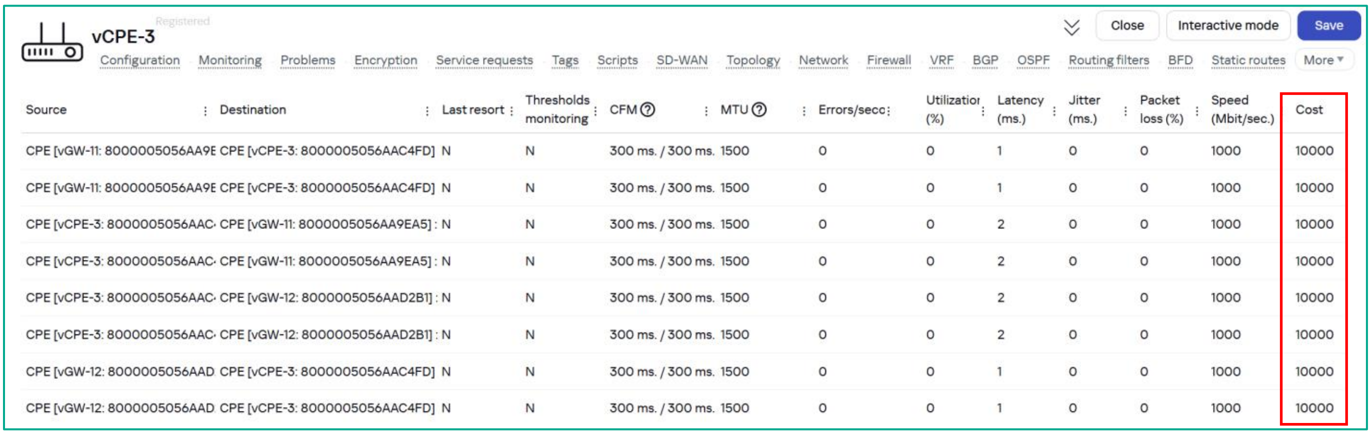


Switch to the **Links** tab.



A list of links established with **vCPE-3** is displayed.

In this scenario, balancing will be performed between links with the same cost, without using multi-weight. The cost value is displayed in the **Cost** column of the **Links** tab. Verify the cost value of the links: the links must have the same cost value for load balancing to work.

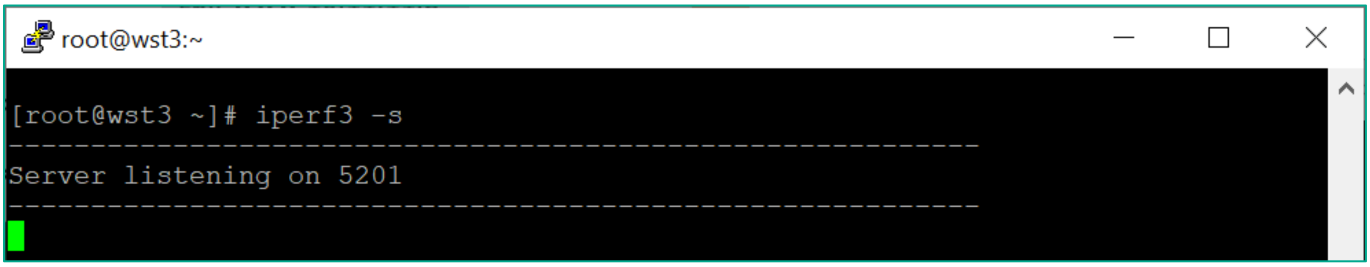


3.1.4. Generate test traffic between wst3 and wst4.

To generate traffic between **vCPE-3** and **vCPE-4, iperf** is used on **wst3** and **wst4**.

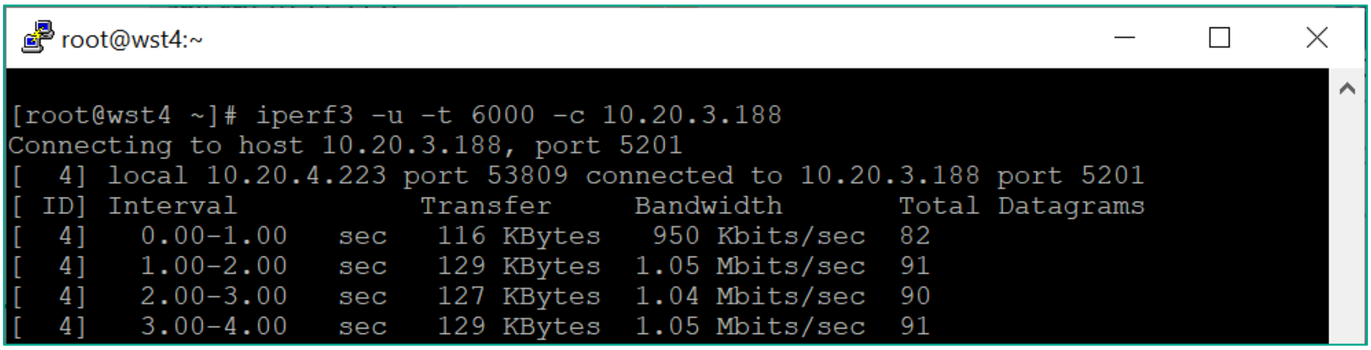
Start the **iperf** server on the **wst3** host:

```
iperf3 -s
```



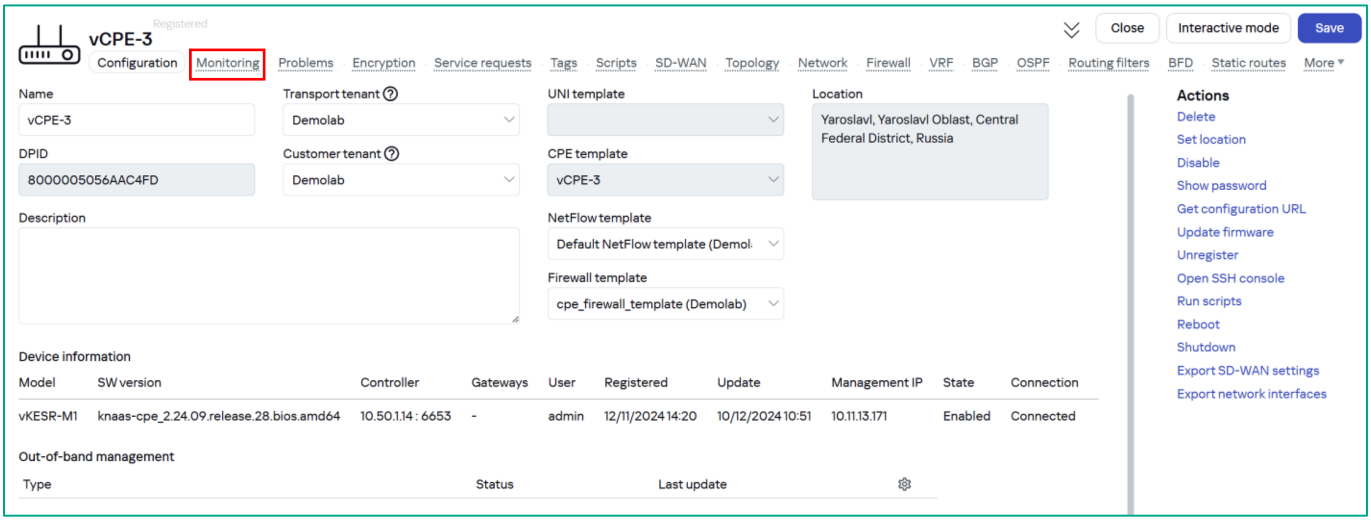
Start the iperf client on the **wst4** host (also check the IP addresses assigned to **wst3** and **wst4** hosts – run `ip a` on workstations):

```
iperf3 -u -t 6000 -c <wst3 IP>
```

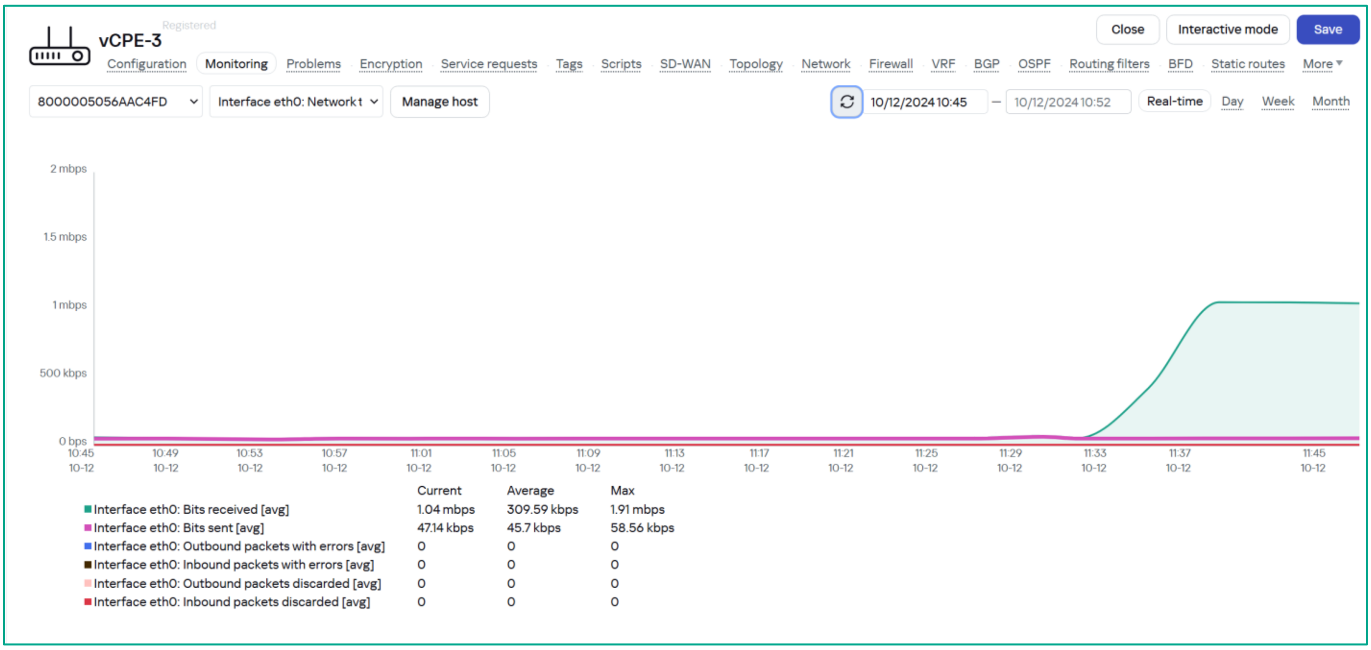


3.1.5. Verify traffic balancing between vCPE-3 WAN interfaces.

Go to the **CPE** menu and select **vCPE-3**, then switch to the **Monitoring** tab.

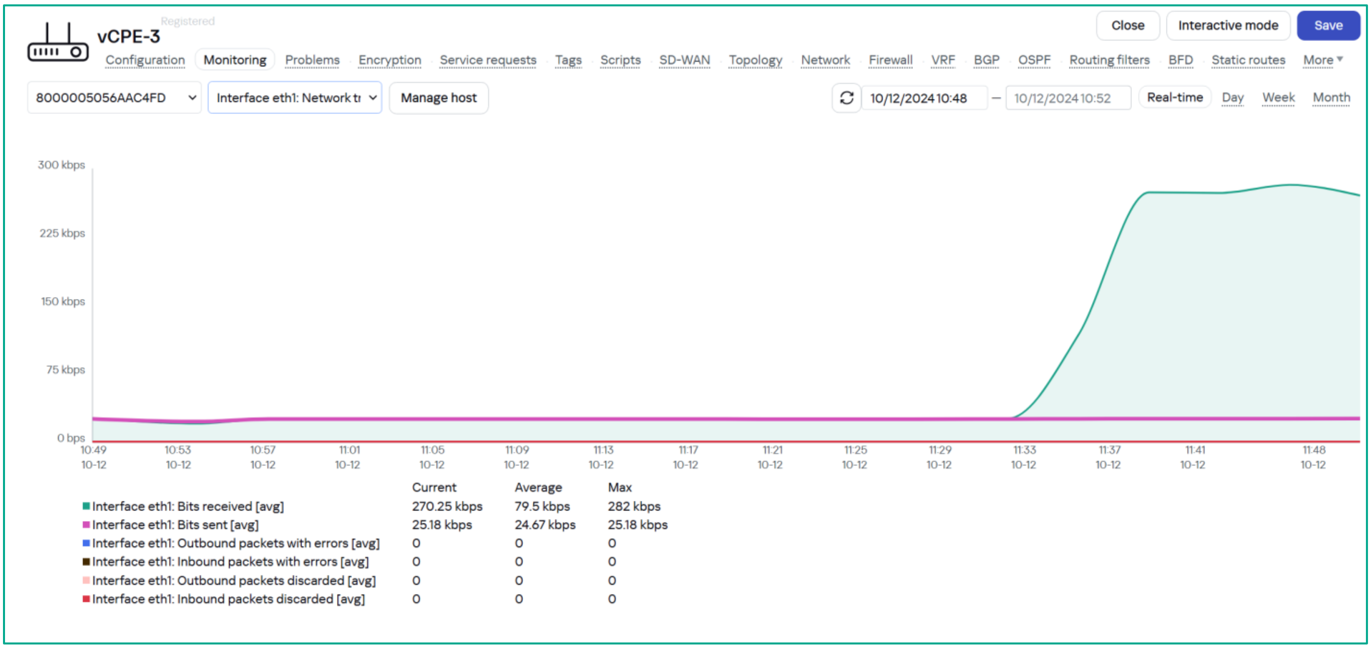


Select **Interface eth0** (sdwan0) and ensure traffic passes through it. A burst will display on the graph labeled "**Interface eth0: Average bit sent**". Wait 10 minutes to collect statistics before displaying data.



Verify the traffic flow through the CPE's second WAN interface.

Select **Interface eth1** and confirm on the graph that network traffic is going through this interface.



As show in the graphs above, traffic is balanced across 2 WAN interfaces.

3.1.6. Restore the settings after the test.

Repeat 3.1.2 to change the transport service balancing mode to **per-flow**.

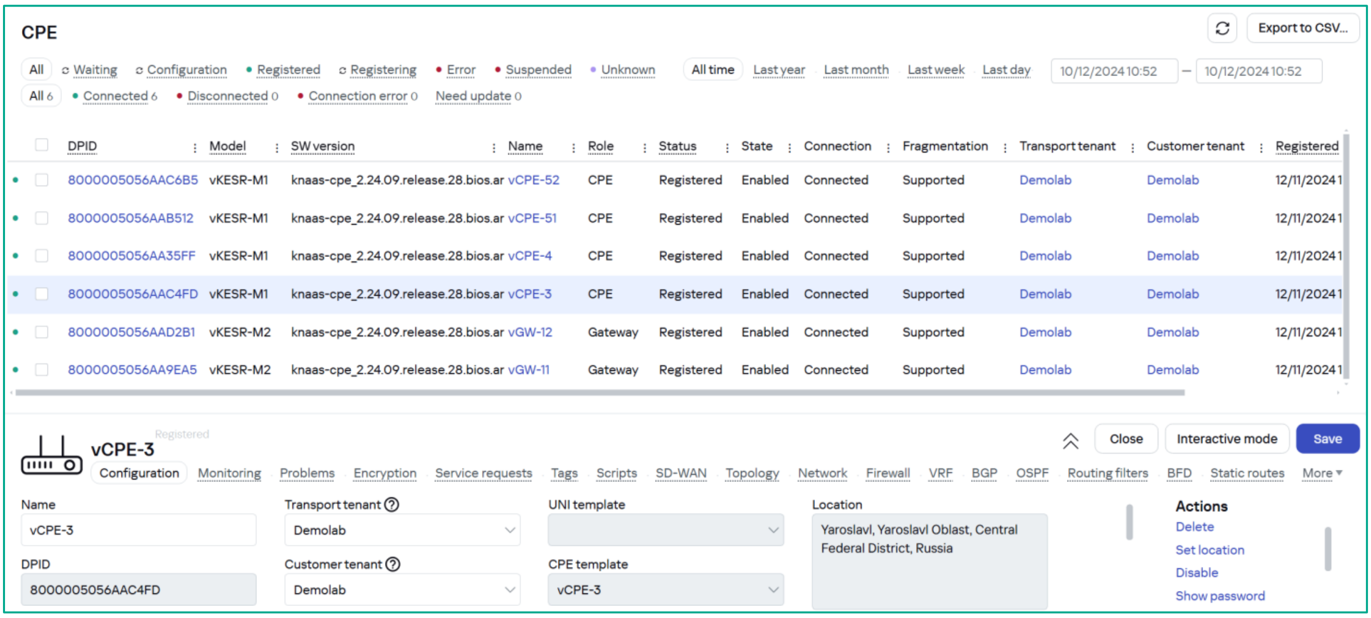
Stop the iperf processes on the **iperf** on the **wst3** and **wst4** hosts, started in 3.1.4 (you can stop it with `Ctrl+Z`).

3.2. **Redundancy with Active/Standby link mode**

This section describes the Active/Standby link redundancy scenario for the vCPE-3 device. The Cost parameter is used to prioritize the WAN interface, on the backup link the parameter value will be increased compared to the primary link. The iperf will be used to generate traffic on workstations wst3 and wst4.The built-in monitoring system will be used to verify the redundancy operation. Demonstration of the backup link operation will be performed by disabling the primary WAN interface of the CPE.

3.2.1. Set cost for the backup links.

Go to the **CPE** menu and select **vCPE-3**.



Switch to the **Links** tab.

The **Links** tab lists the links established by the selected CPE with adjacent CPE devices. The **Source** and **Destination** columns display the source and destination CPEs respectively for each unidirectional link. The port number indicates the WAN interface number of the CPE device, assigned consecutively starting with port 4800, with one for each WAN interface. Port 4800 corresponds to WAN interface sdwan0 (eth0), and port 4801 corresponds to WAN interface sdwan1 (eth1).

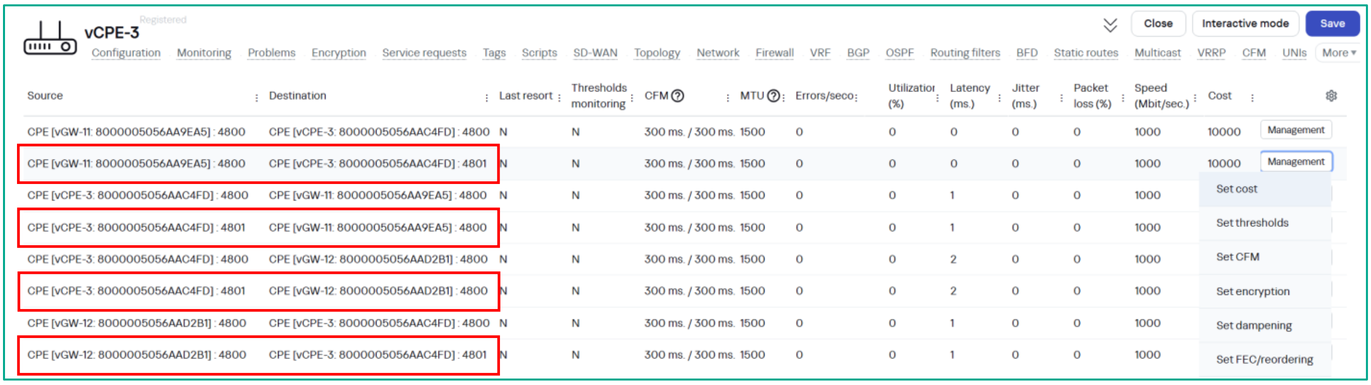
In the SD-WAN solution, the default topology is hub and spoke, so traffic between CPEs flows through the gateway devices. In this scenario, the cost of links going through the redundant WAN interface (**sdwan1/eth1**) of **vCPE3** between **vCPE-3** and **vGW-11** / **vGW-12** gateways should be increased.

Find all links between **vCPE-3** and **vGW-11** / **vGW-12** that are established through the second WAN interface of **vCPE-3** (port **4801**):

- **vCPE-3:4801 -** vGW-11:4800
- **vCPE-3:4801 -** vGW-12:4800
- vGW-11:4800 **- vCPE-3:4801**
- vGW-12:4800 **- vCPE-3:4801**

Change the cost for the found links (by default, the cost depends on the **Maximum rate** value of the SD-WAN interfaces, in the PoC it is 1000).

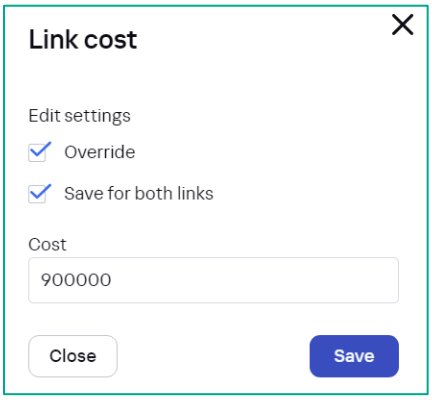
Click **Management → Set cost**.



Set link cost:

- Check **Override**.
- **Cost**: **900000**.
- Check **Save for both links**.

Click **Save**.



Note: It is possible to control the cost of links by changing the **Maximum rate** value in the SD-WAN interfaces settings. But the same value also affects the shaper configured for the outgoing traffic of the SD-WAN interfaces.

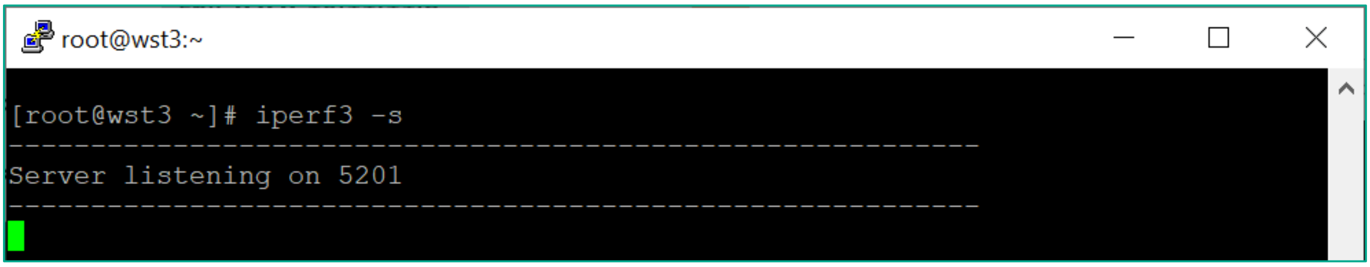
kaspersky

3.2.2. Generate traffic between wst3 and wst4.

iperf is used to generate traffic between **vCPE-3** and **vCPE-4** on **wst3** and **wst4** workstations.

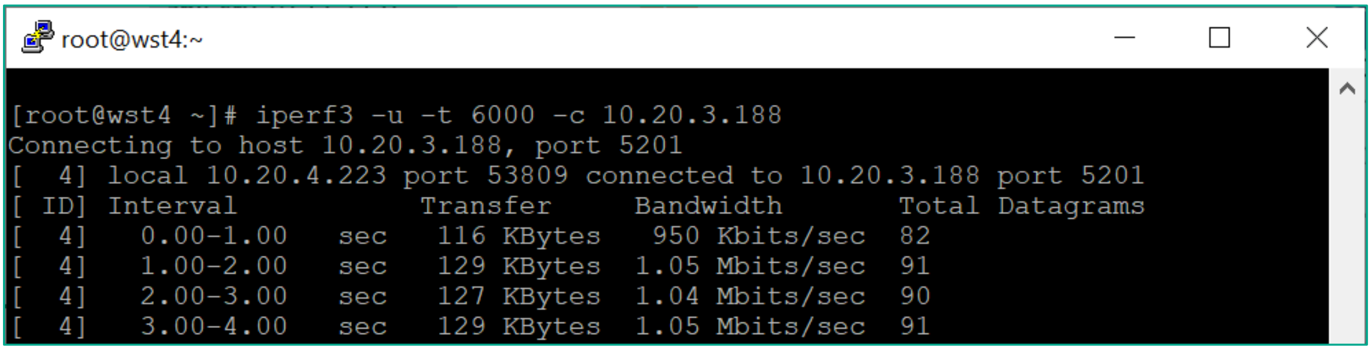
Start the **iperf3** server on the **wst3** host:

```
iperf3 -s
```



Start the **iperf3** client on the **wst4** host (also check the IP addresses assigned to **wst3** and **wst4** hosts – run `ip a` on workstations):

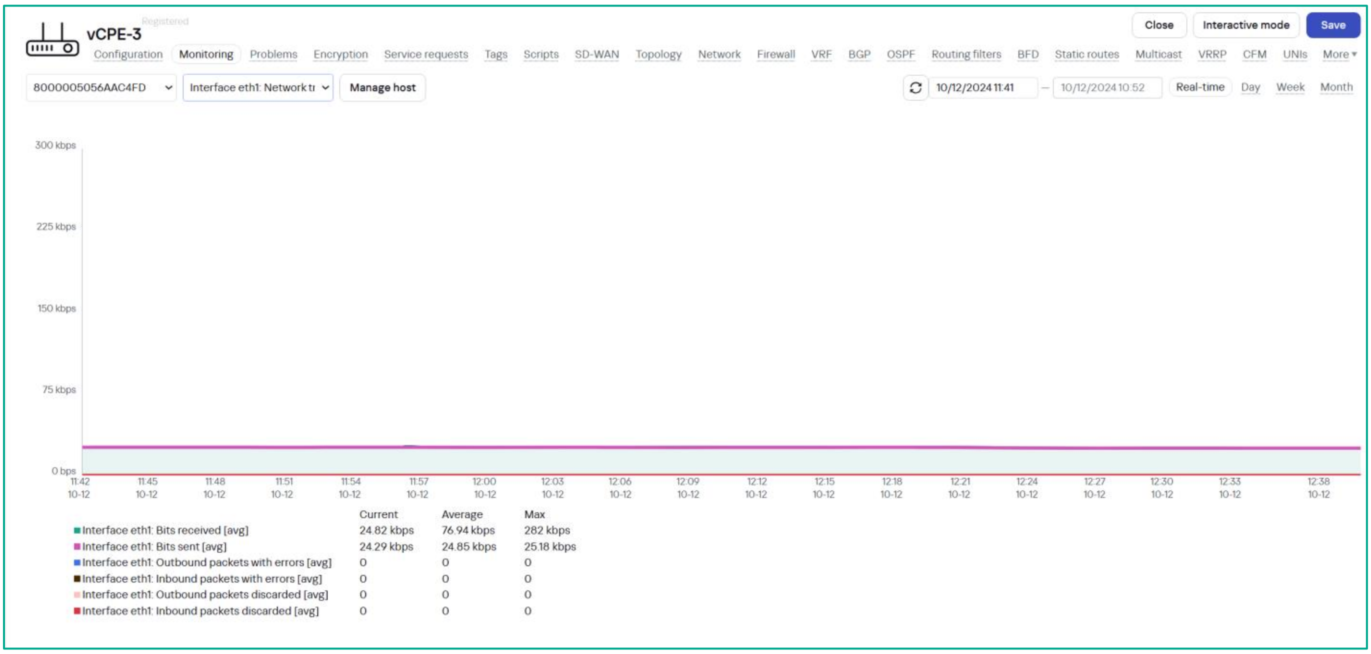
```
iperf3 -u -t 6000 -c <wst3 IP>
```



3.2.3. Verify traffic statistics of the vCPE-3 WAN interfaces in the monitoring system.

Go to the **CPE** menu, select **vCPE-3**. Open the **Monitoring** tab. Select the **eth0** interface and verify on the graph that the traffic passes through it.

Select interface **eth1** and verify that there is no network traffic passing through it by checking the graph labeled **Interface eth1: Bit sent[avg]**.



3.2.4. Verify WAN interfaces redundancy operation.

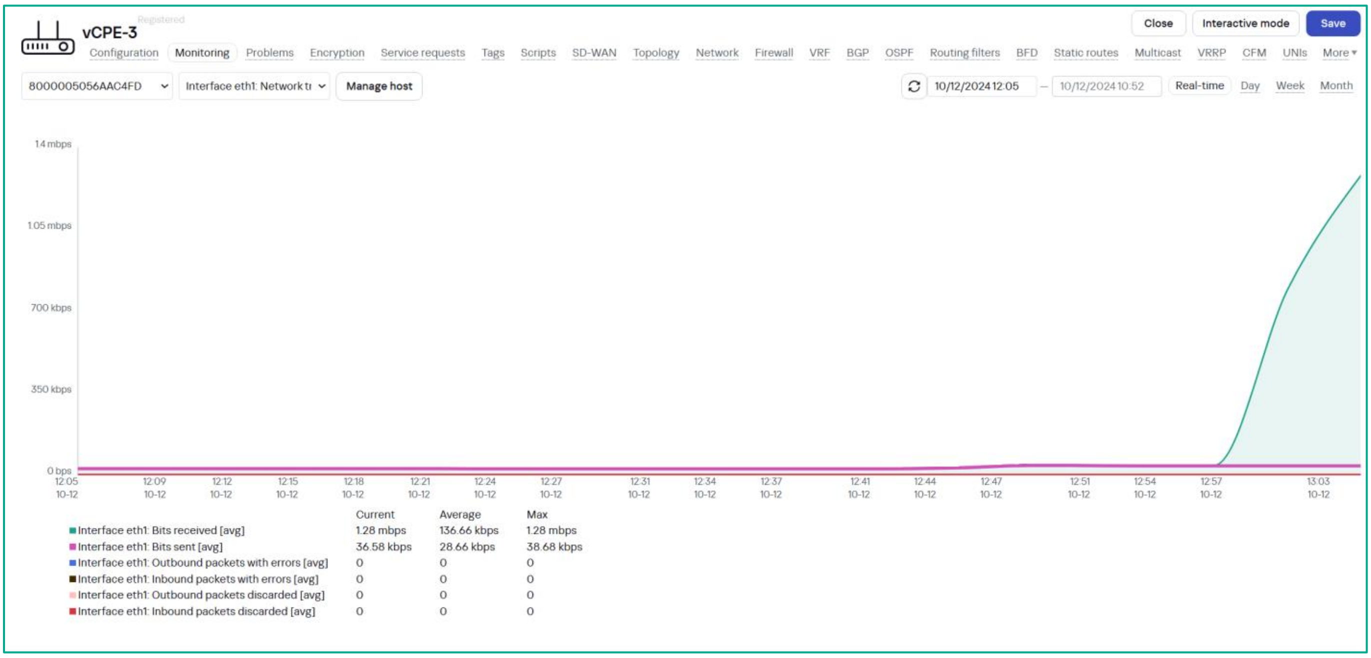
Emulate the failure of the primary WAN interface:

Connect to the **isp** router and disable the network interface to which the **sdwan0** (**eth0**) network interface of the **vCPE-3** device is connected:

```
ifconfig ens161 down
```

It may be necessary to restart the **iperf3** client on **wst-3**: repeat 3.2.2.

Go to the **CPE** menu and select **vCPE-3**. Open the **Monitoring** tab. Select the **eth1** interface and verify on the graph that traffic has switched to this network interface.



3.2.5. Restore the settings after the test is completed.

Enable the network interface on the **isp** host disabled in 3.2.4:

```
ifconfig ens161 up
```

Return the **vCPE-3** links **cost** values changed in 3.2.1 to the default ones. Stop the **iperf3** processes on **wst3** and **wst4** started in 3.2.2 (you can stop it with `Ctrl+Z`).

3.3. **Packet loss overcome with packet duplication in broadcast mode**

Kaspersky SD-WAN provides protection against traffic failures through simultaneous use of available network channels. To achieve additional fault tolerance, a broadcast balancing mode is supported – in this mode, copies of packets are sent to all links simultaneously, effectively eliminating losses.

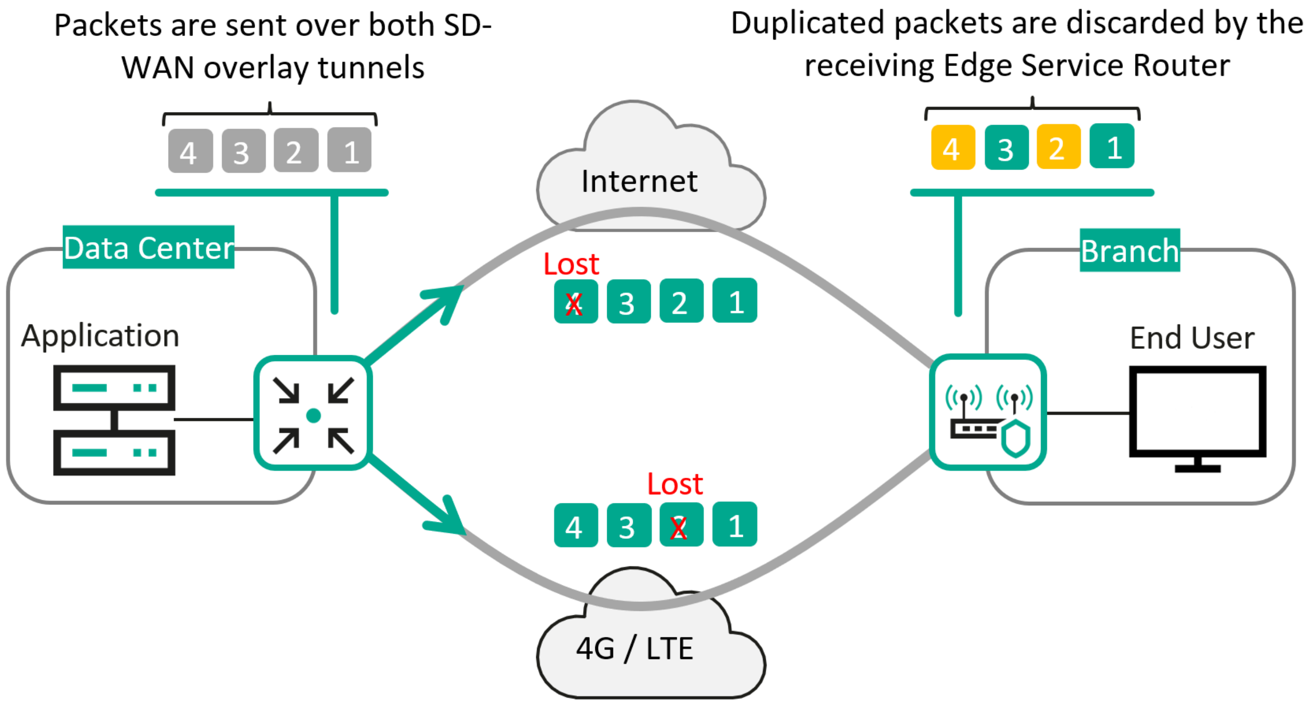


Figure 3.3.1 Packet Duplication

This section covers a redundancy scenario between vCPE-3 links. To achieve redundancy, broadcast mode is used. In this mode, the CPE sends copies of packets simultaneously over all available links.

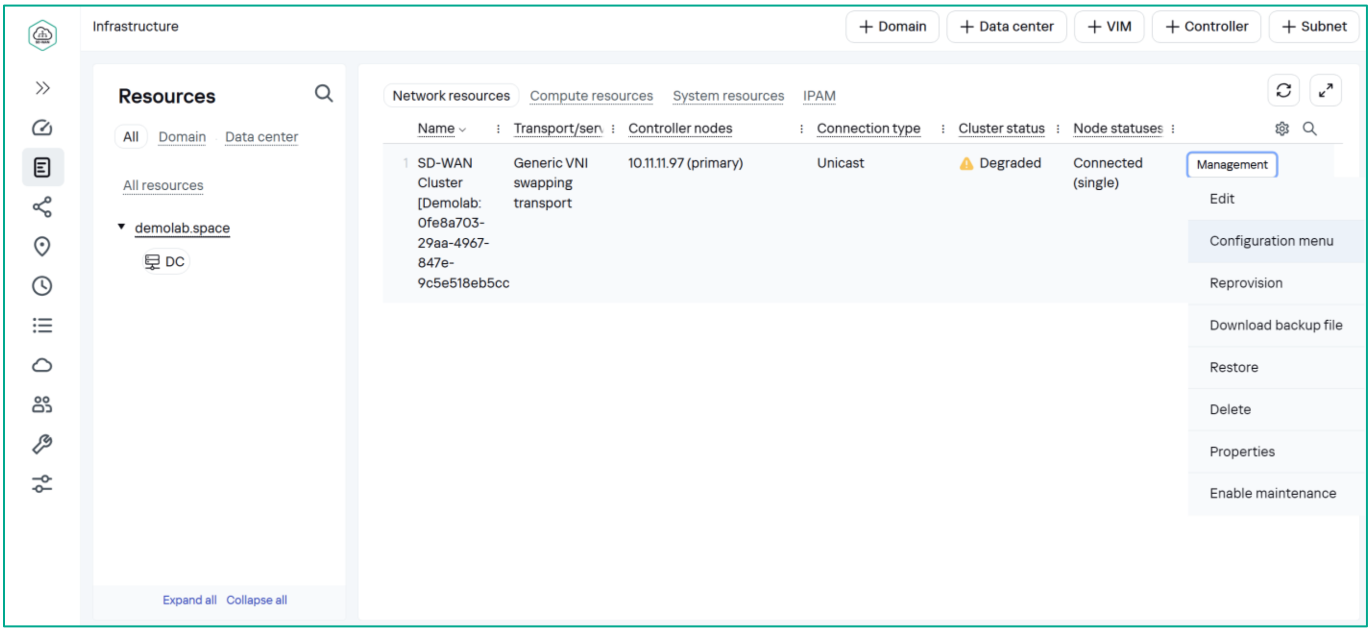
ICMP ping is used to generate traffic between hosts wst3 and srv1. To verify the redundancy is working, tcpdump on vCPE-3 will be used.

3.3.1. Set broadcast balancing mode for transport service.

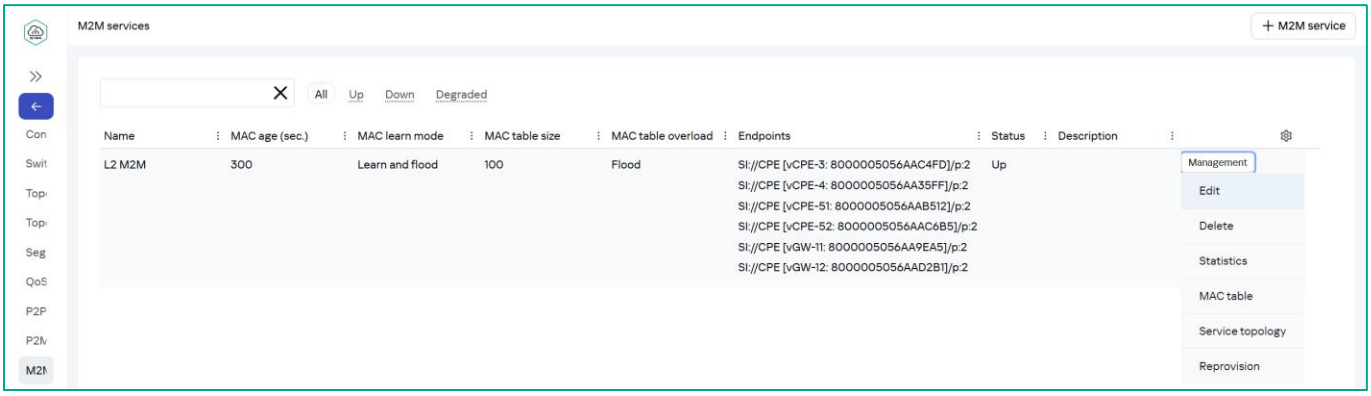
Available balancing modes:

- Per-flow (session) balancing. During transmission, flows are distributed evenly across links.
- Per-packet balancing. Packets are distributed evenly across links during transmission.
- Broadcast. Packets are transmitted simultaneously to all links to eliminate losses.

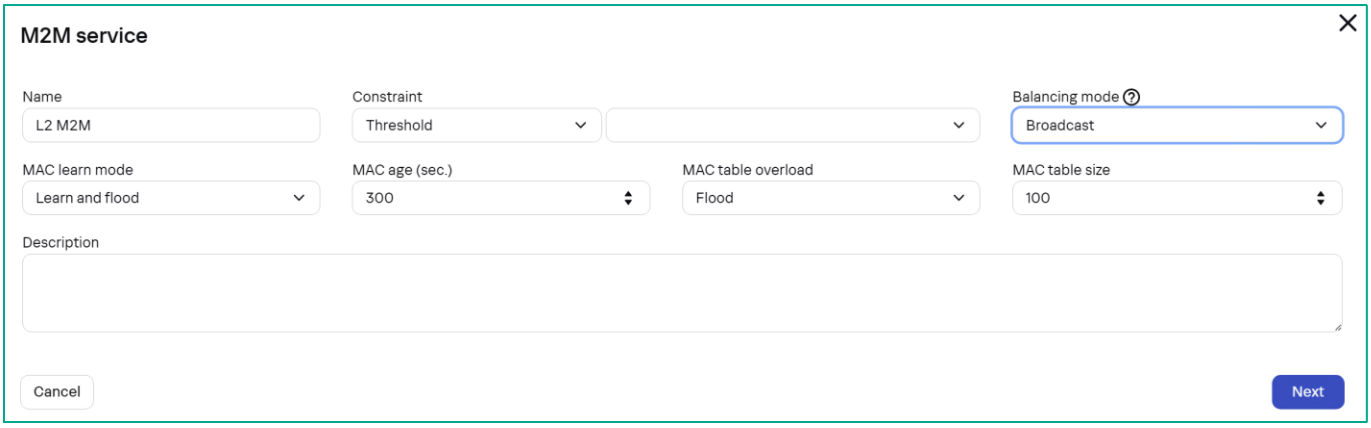
To configure load balance for transport service, go to **Infrastructure → SD-WAN cluster → Configuration menu**.



Go to the **M2M services** menu. Select M2M transport service, then click **Management → Edit**.



Set **Balancing mode**: **Broadcast**.



Click **Next**, **Next** then **Save**.

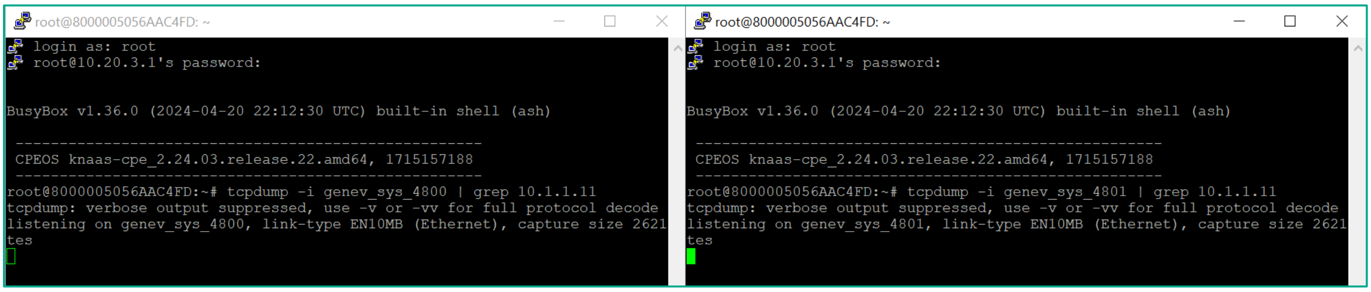
3.3.2. Verify broadcast balancing mode operation.

Open 2 SSH sessions to **vCPE-3**.

Start **tcpdump** for tunnel interfaces: in session 1 for **genev_sys_4800**, in session 2 for **genev_sys_4801** interface:

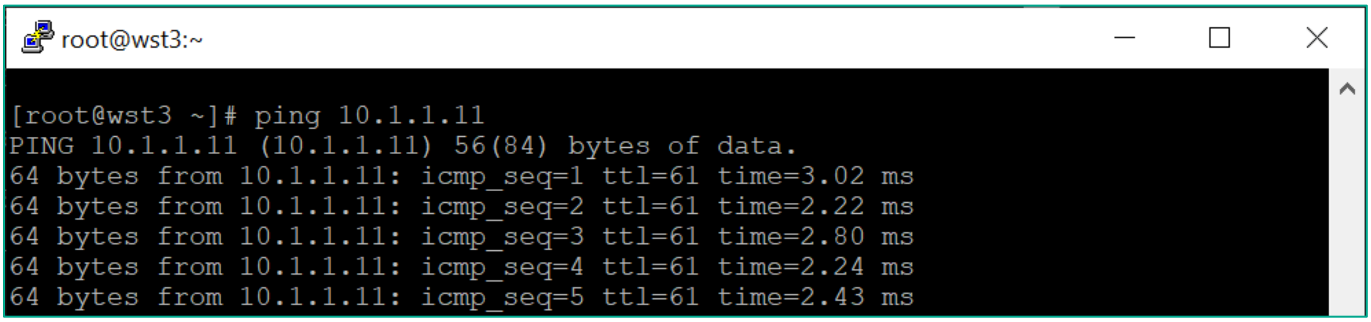
```
tcpdump -i genev_sys_4800 | grep <srv1 IP>
tcpdump -i genev_sys_4801 | grep <srv1 IP>
```

genev_sys – are the CPE tunnel Interfaces. Each port number corresponds to a WAN interface number. The numbers are assigned consecutively, starting with port 4800, one for each WAN interface. Port 4800 is designated for WAN interface sdwan0 (eth0), while port 4801 corresponds to WAN interface sdwan1 (eth1).

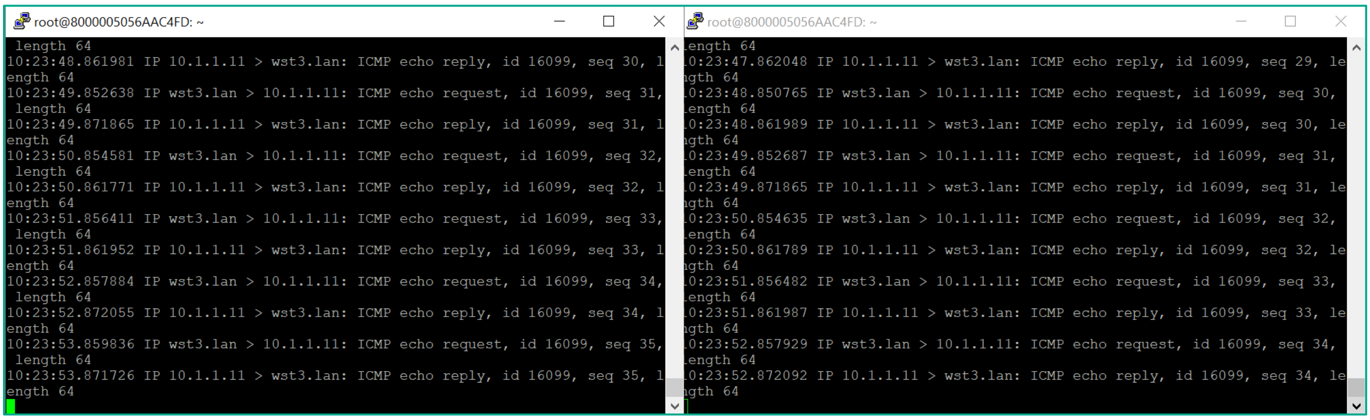


Start ICMP **ping** from the **wst3** host to **srv1**:

```
ping <srv1 IP>
```



ICMP packets will appear in the tcpdump output on vCPE-3. You can see that a copy of the packets was sent to each interface (packets have the same **sequence number**).



3.3.3. Restore the settings after the test is completed.

Repeat step 3.3.1 to change load balance mode to **per-flow**.

Stop ICMP **ping** on **wst3** and **tcpdump** on **vCPE-3**, started in 3.3.2 (you can stop it with `Ctrl+Z`).

3.4. **Improving network channels reliability through Forward Error Correction**

The Forward Error Correction (FEC) functionality reduces the loss of traffic packets in communication channels, especially for UDP applications, and the number of retransmissions, which lead to delays, and recovers received data on the CPE device. Data recovery is provided by redundant encoding of the data stream on the device on the sending side.

The sender CPE encodes the stream of traffic packets egressing into the link, adding redundant packets. The use of encoding on the sending and receiving sides may cause delays due to extra data processing. You can configure the degree of redundancy in the settings of the SD-WAN controller or when you enable FEC.

The receiving CPE device buffers traffic packets received through the link and decodes them, recovering lost packets, if possible. The general diagram of FEC operation is shown in the figure below.

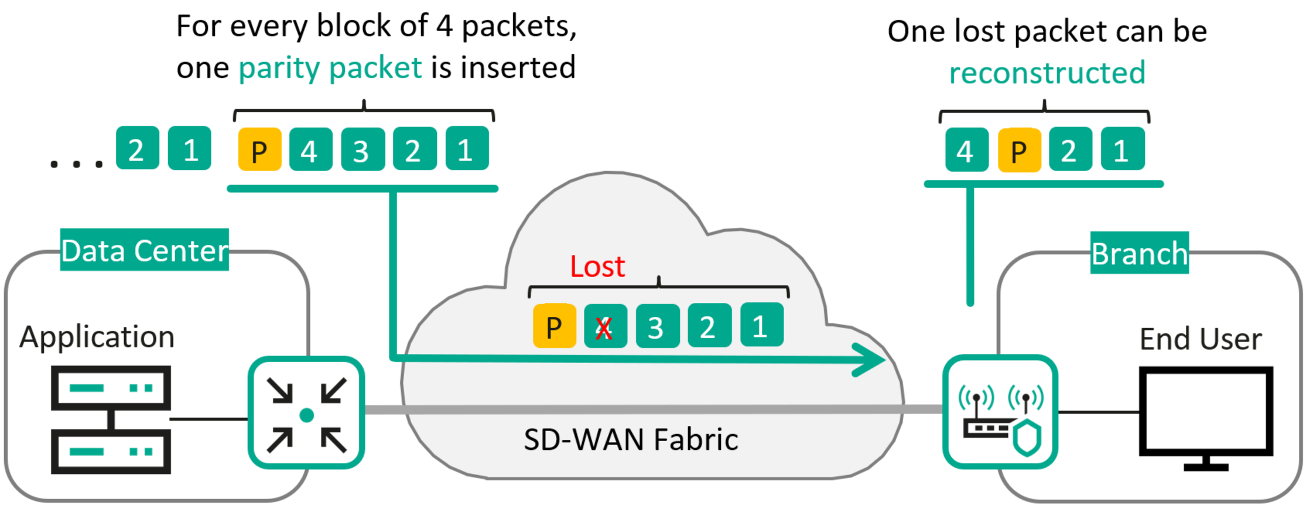


Figure 3.4.1 Forward Error Correction (FEC)

The use of FEC can reduce the impact of high packet loss ratio on data links, especially for UDP applications, and reduces the number retransmissions for TCP sessions. It is recommended to use FEC on so-called noisy links to reduce the packet loss ratio and increase the speed of TCP connections.

For more information, please refer to Kaspersky SD-WAN Online Help:
https://support.kaspersky.com/help/SD-WAN/2.3/en-US/245033.htm

This section covers a scenario with channel loss emulation, measuring links quality and enabling FEC to recover lost packets. The test traffic will be generated between hosts wst3 and srv1 using ICMP ping.

Packet loss emulation will be performed on the isp host with Linux Traffic Control (TC).

3.4.1. Generate test traffic between wst3 and srv1.

Start icmp **ping** from the **wst3** host to **srv1**:

```
ping <srv1 IP>
```

root@wst3:~

```
[root@wst3 ~]# ping 10.1.1.11
PING 10.1.1.11 (10.1.1.11) 56(84) bytes of data.
64 bytes from 10.1.1.11: icmp_seq=1 ttl=61 time=3.02 ms
64 bytes from 10.1.1.11: icmp_seq=2 ttl=61 time=2.22 ms
64 bytes from 10.1.1.11: icmp_seq=3 ttl=61 time=2.80 ms
64 bytes from 10.1.1.11: icmp_seq=4 ttl=61 time=2.24 ms
64 bytes from 10.1.1.11: icmp_seq=5 ttl=61 time=2.43 ms
```

3.4.2. Emulate packet loss on the **isp** host using TC.

For the test, you must enable packet loss emulation on the **isp** host network interface to which the **sdwan0** (**eth0**) interface of **vCPE-3** is connected.

Connect to the **isp** host and execute:

```
tc qdisc add dev ens161 root netem delay 1ms 0ms limit 1250000 loss 5%
```

This command creates **5%** packet **loss**. The **delay** parameter configures a delay of **1ms** with a spread of **0ms**, **limit** - allocates a buffer of **1250000** bytes for TC data processing.

Check the applied settings using the following command:

```
tc qdisc show
```

root@isp:~

```
[root@isp ~]# tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8001: dev ens161 root refcnt 2 limit 1250000 delay 1.0ms loss 5%
qdisc pfifo_fast 0: dev ens192 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1
1 1 1 1
qdisc pfifo_fast 0: dev ens193 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1
1 1 1 1
qdisc pfifo_fast 0: dev ens224 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1
1 1 1 1
qdisc pfifo_fast 0: dev ens225 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1
1 1 1 1
qdisc pfifo_fast 0: dev ens256 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1
1 1 1 1
qdisc pfifo_fast 0: dev ens257 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1
1 1 1 1
[root@isp ~]#
```

Note: By default, traffic is balanced on a per-flow basis. Therefore, if a packet flow traverses another interface, loss emulation will not affect the traffic. Per-flow mode should be used in this scenario.

As you can see below by the ICMP **sequence numbers**, packet losses are present, missed replies are visible (missed sequence **281**, **290**, **293**).

```
root@wst3:~
64 bytes from 10.1.1.11: icmp_seq=74 ttl=61 time=10.8 ms
64 bytes from 10.1.1.11: icmp_seq=75 ttl=61 time=18.8 ms
64 bytes from 10.1.1.11: icmp_seq=76 ttl=61 time=6.57 ms
64 bytes from 10.1.1.11: icmp_seq=77 ttl=61 time=4.77 ms
64 bytes from 10.1.1.11: icmp_seq=78 ttl=61 time=13.1 ms
64 bytes from 10.1.1.11: icmp_seq=80 ttl=61 time=41.7 ms
64 bytes from 10.1.1.11: icmp_seq=81 ttl=61 time=30.4 ms
64 bytes from 10.1.1.11: icmp_seq=82 ttl=61 time=59.6 ms
64 bytes from 10.1.1.11: icmp_seq=83 ttl=61 time=27.7 ms
64 bytes from 10.1.1.11: icmp_seq=84 ttl=61 time=45.8 ms
64 bytes from 10.1.1.11: icmp_seq=85 ttl=61 time=24.8 ms
64 bytes from 10.1.1.11: icmp_seq=86 ttl=61 time=42.7 ms
64 bytes from 10.1.1.11: icmp_seq=87 ttl=61 time=10.7 ms
64 bytes from 10.1.1.11: icmp_seq=88 ttl=61 time=38.8 ms
64 bytes from 10.1.1.11: icmp_seq=89 ttl=61 time=36.7 ms
64 bytes from 10.1.1.11: icmp_seq=90 ttl=61 time=34.8 ms
64 bytes from 10.1.1.11: icmp_seq=92 ttl=61 time=32.6 ms
64 bytes from 10.1.1.11: icmp_seq=94 ttl=61 time=40.6 ms
64 bytes from 10.1.1.11: icmp_seq=95 ttl=61 time=48.8 ms
64 bytes from 10.1.1.11: icmp_seq=97 ttl=61 time=27.5 ms
64 bytes from 10.1.1.11: icmp_seq=98 ttl=61 time=46.6 ms
64 bytes from 10.1.1.11: icmp_seq=100 ttl=61 time=44.6 ms
64 bytes from 10.1.1.11: icmp_seq=101 ttl=61 time=62.7 ms
```

If the statistics do not show packet losses, it means that the traffic goes through an interface where loss emulation is not applied and it is necessary to apply emulation to another interface on the **isp** host (to **eth1** on **vCPE-3**):

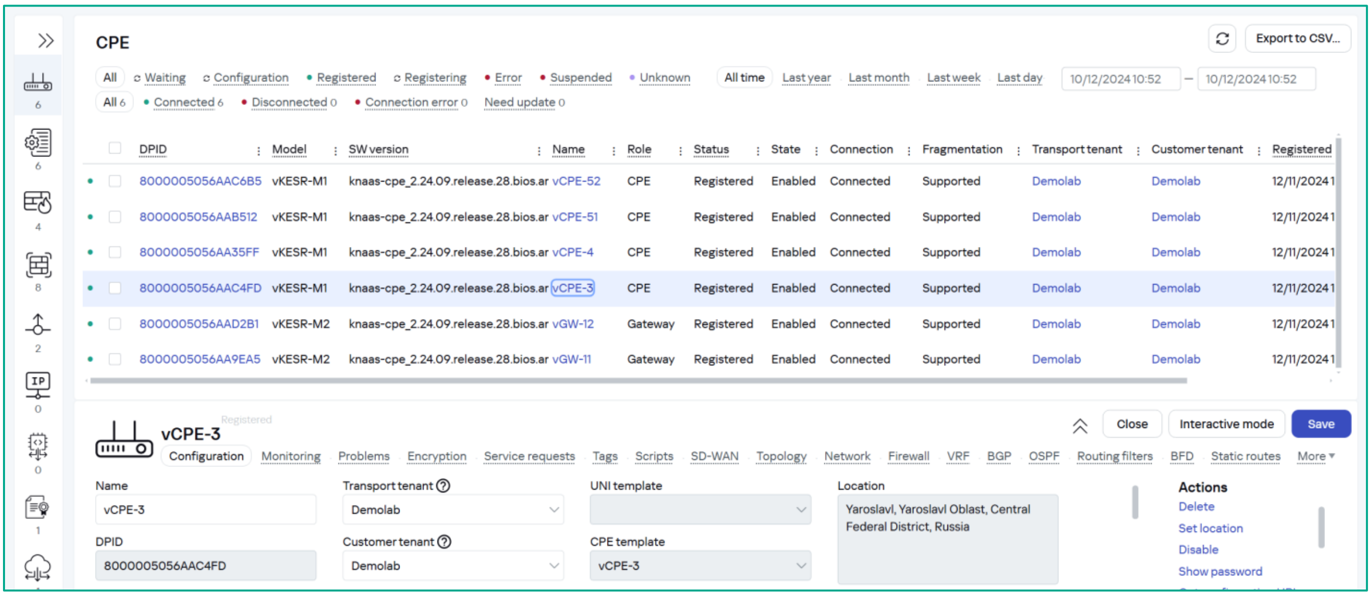
```
tc qdisc add dev ens193 root netem delay 1ms 0ms limit 1250000 loss 5%
```

And remove loss emulation from the previous network interface (to **eth0** on **vCPE-3**):

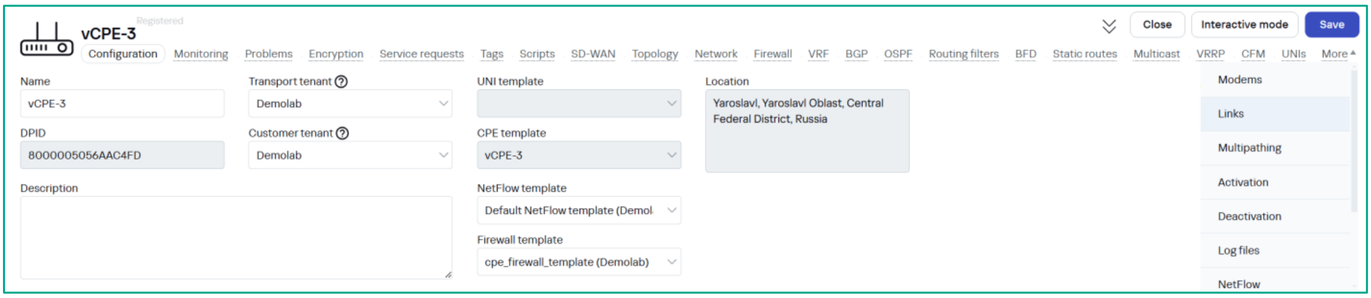
```
tc qdisc del dev ens161 root
```

3.4.3. Enable packet loss monitoring for the vCPE-3 links.

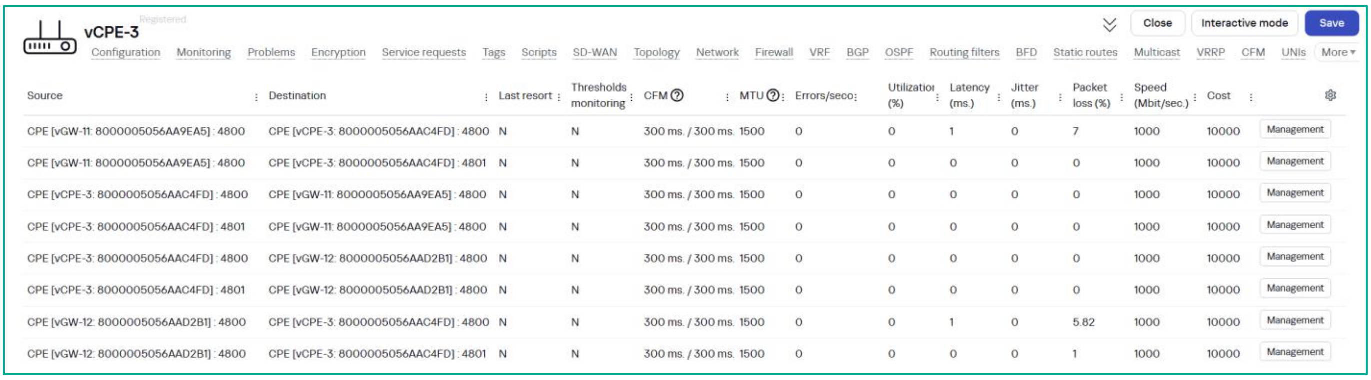
Go to the **CPE** menu, then select **vCPE-3**.



Switch to the **Links** tab.



A list of links established with **vCPE-3** is displayed.

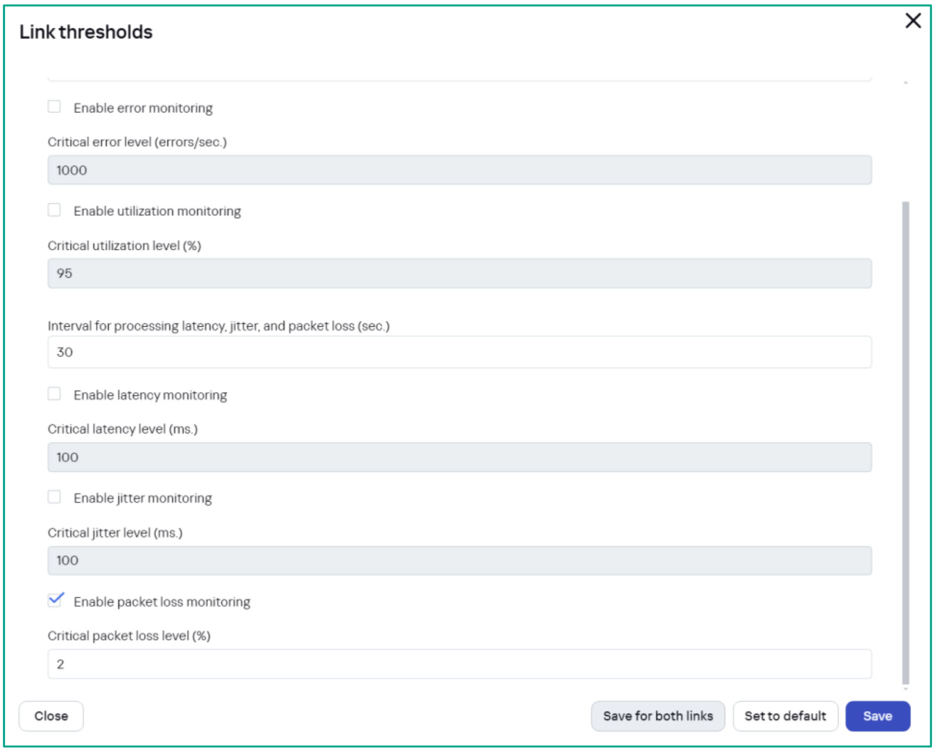


For each link click **Management → Set thresholds**.

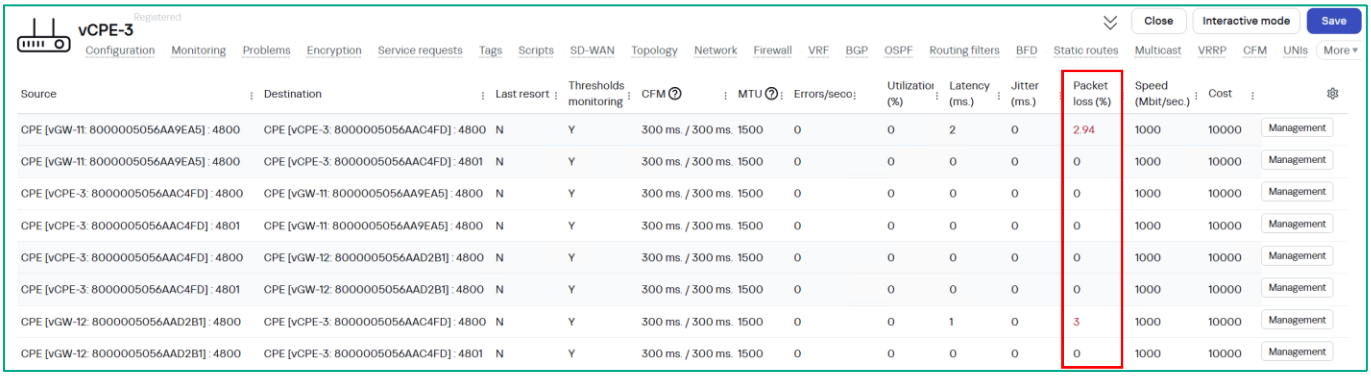
Set link monitoring settings:

- Check **Enable tunnel thresholds monitoring**.
- **Enable packet loss monitoring → Critical packet loss level: 2%**.

Click **Save for both links**.



After the settings are applied, the loss statistics for links will be displayed. The values of measured parameters that do not meet the thresholds set earlier will be highlighted in red. Since the delay was emulated on the **sdwan0** (**eth0**) **vCPE-3** interface, packet loss is observed on the corresponding links from the **vGW-11** and **vGW-12** passing through this interface.

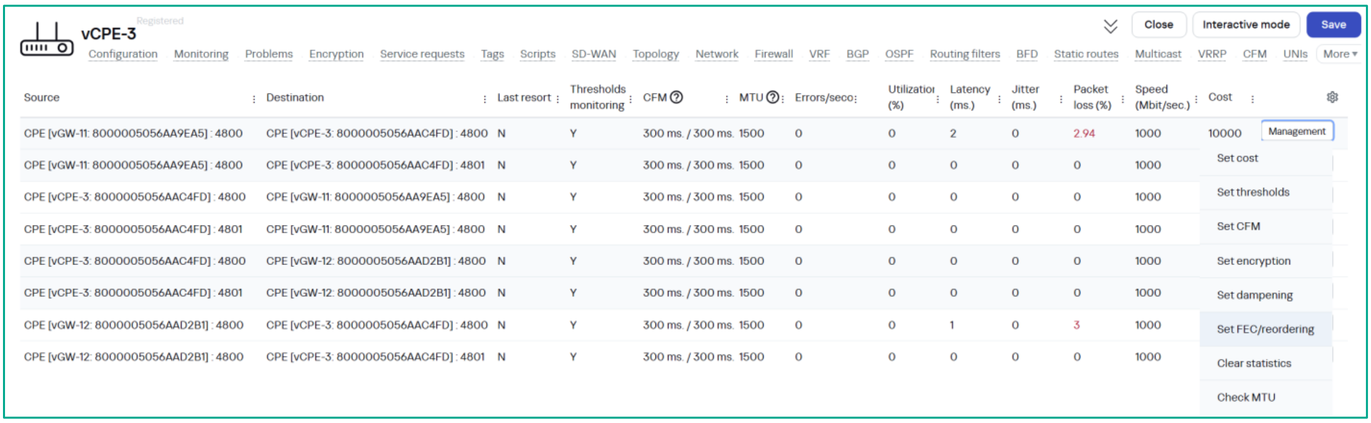


Source	Destination	Last resort	Thresholds monitoring	CFM	MTU	Errors/seco	Utilizatio (%)	Latency (ms.)	Jitter (ms.)	Packet loss (%)	Speed (Mbit/sec.)	Cost	
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	2	0	2.94	1000	10000	Management
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	1	0	3	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management

kaspersky

3.4.4. Enable FEC on lossy vCPE-3 links.

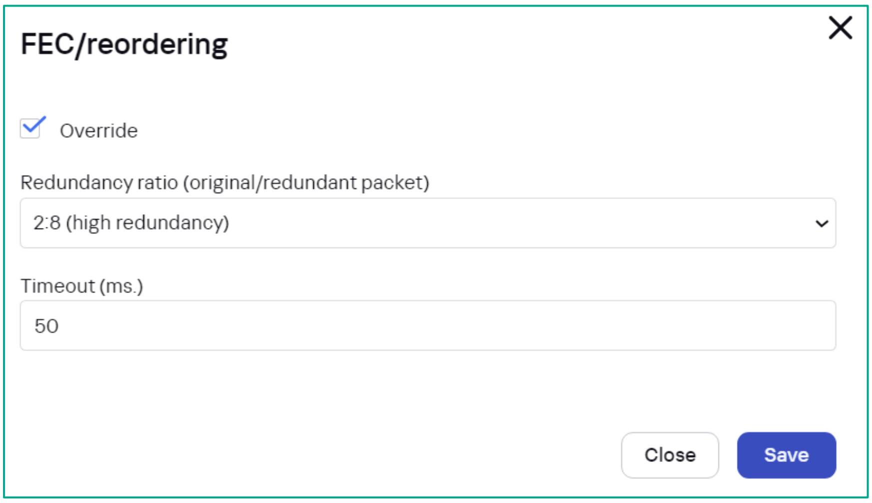
Click **Management → Set FEC/reordering** for all links with packet loss.



Set FEC settings:

- Check **Override**.
- **Redundancy ratio**: **2:8**.
- **Timeout**: **50**.

Click **Save**.

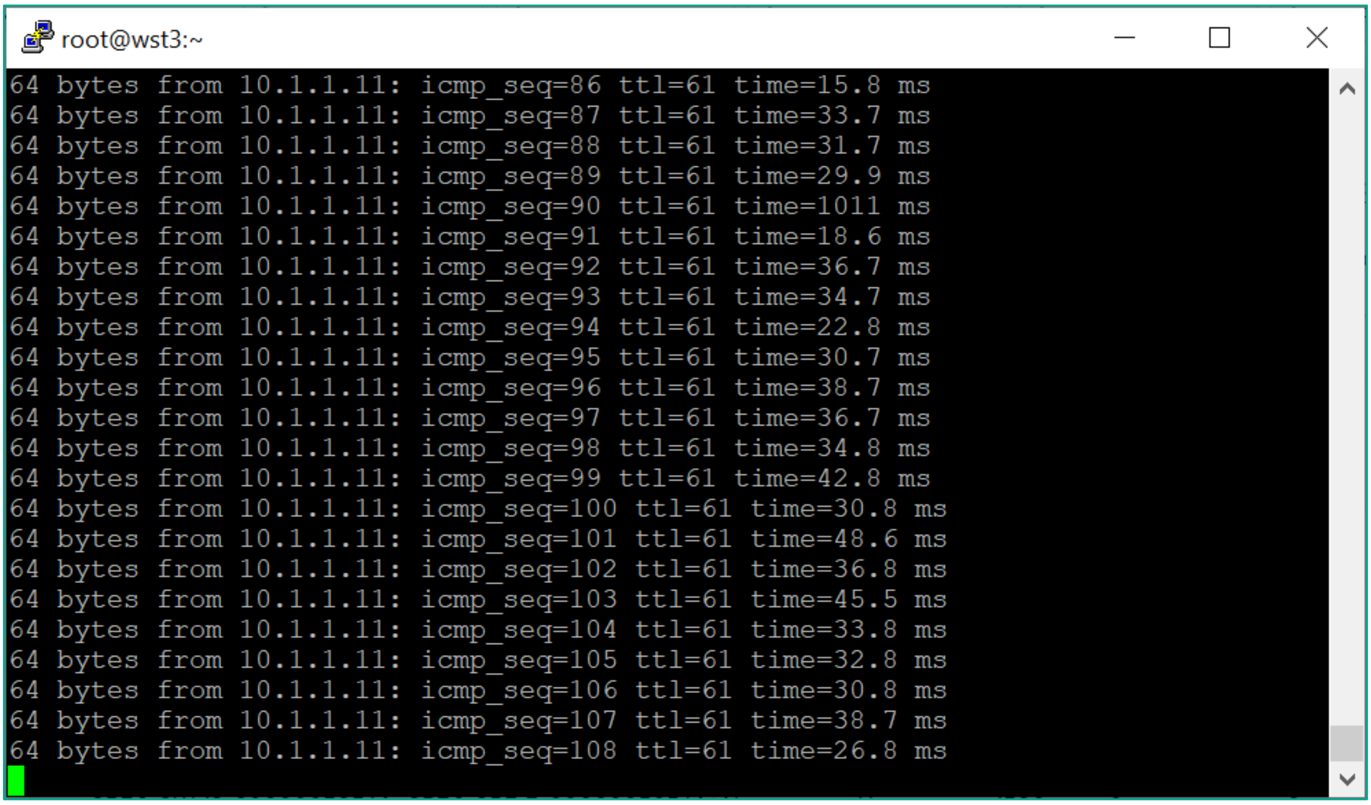


kaspersky

3.4.5. Verify FEC operation in ping statistics.

Check the ICMP **ping** statistics on host **wst3** for the missing packets.

The statistics show that all ICMP packets are successfully received: there are no missed ICMP replies (no missed sequence numbers). Packets are successfully restored with FEC.



3.4.6. Restore the settings after the test is completed.

Repeat step 3.4.3 to disable link **packet loss** monitoring on **vCPE-3**.

Repeat step 3.4.4 to disable **FEC** for **vCPE-3** links.

Stop ICMP ping on **wst3**, started in 3.4.1 (you can stop it with `Ctrl+Z`).

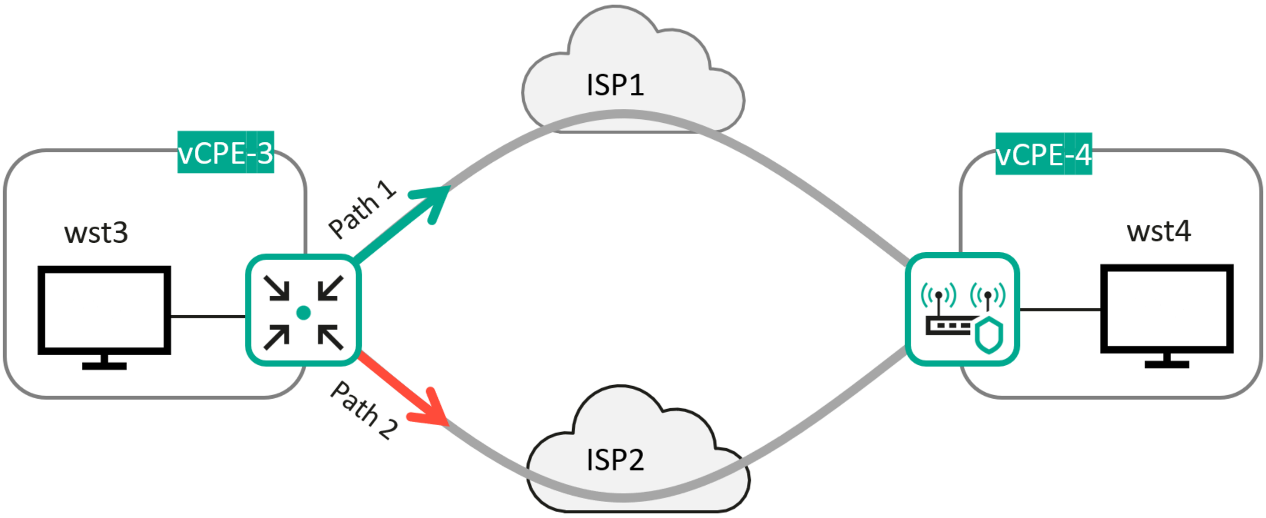
Disable packet loss emulation on **isp** host:

```
tc qdisc del dev ens161 root netem

tc qdisc del dev ens193 root netem
```

3.5. Link quality monitoring (Jitter, Latency, Packet Loss) and traffic management

The SD-WAN solution allows you to measure link parameters (jitter, latency, packet loss) and modify the traffic paths based on the specified parameters, e.g. to minimize latency. Links parameters are measured with additional Type-Length Value (TLV) fields within GENEVE packets headers.



	Jitter	Packet Loss	Latency
Path 1	71 ms	0 %	297
Path 2	4 ms	0 %	15

Figure 3.5.1 Links monitoring

For more information, please refer to Kaspersky SD-WAN Online Help: https://support.kaspersky.com/help/SD-WAN/2.3/en-US/244988.htm

This section describes a scenario where delay and jitter are measured on links, constraints are set, and traffic is redirected to the links that satisfy the delay and jitter requirements. Test traffic will be generated between wst3 and wst4 hosts using iperf.

Delay and jitter emulation will be done on the isp host with Linux Traffic Control.

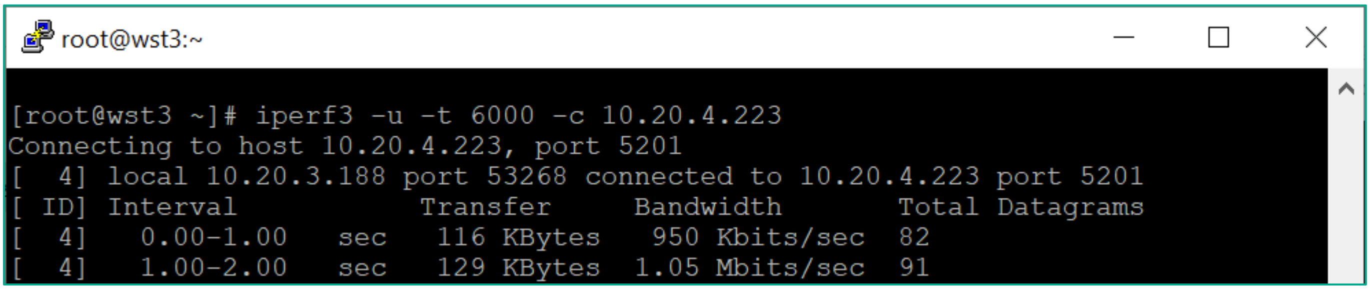
Restrictions will be created for the transport service in order to exclude links that do not meet the specified jitter and delay parameters.

For latency monitoring to work correctly, all CPE devices and gateways must have access to NTP servers and the system time on the devices must be synchronized.

3.5.1. Generate test traffic between wst3 and wst4.

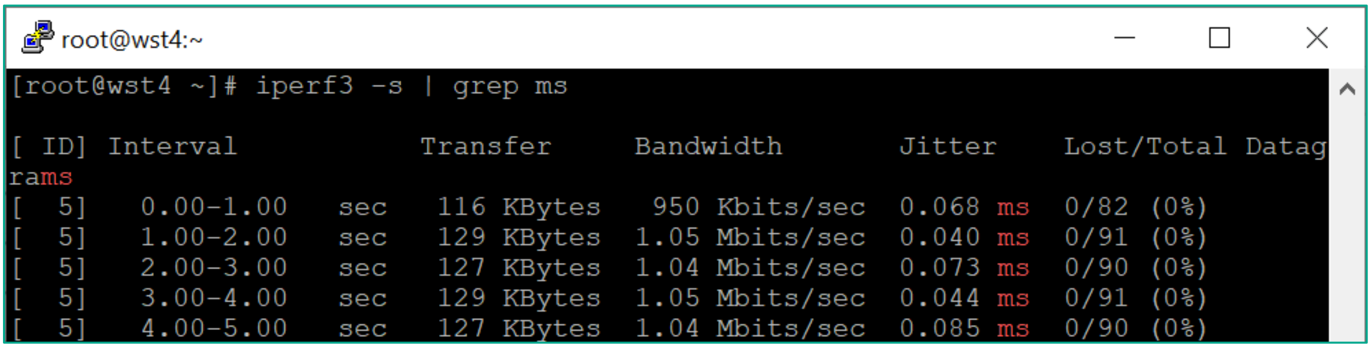
Start the iperf server on the **wst4** host:

```
iperf3 -s | grep ms
```



Start the **iperf** on the **wst3** host:

```
iperf3 -u -t 6000 -c <wst4 IP address>
```



3.5.2. Emulate delay and jitter on the interface to the vCPE-3 using TC.

For the test, you must enable delay and jitter emulation on the **isp** host network interface to which the **sdwan0** (**eth0**) interface of **vCPE-3** is connected.

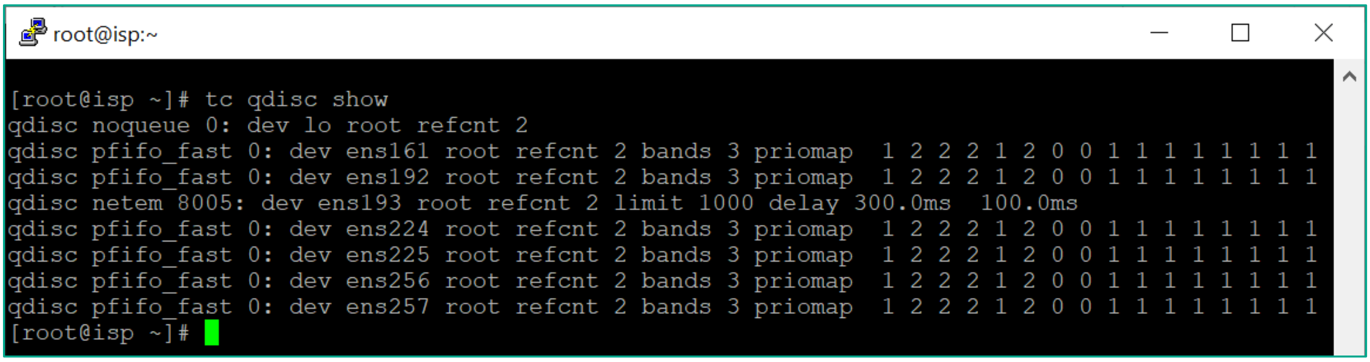
Connect to the **isp** host and execute:

```
tc qdisc add dev ens193 root netem delay 300ms 100ms
```

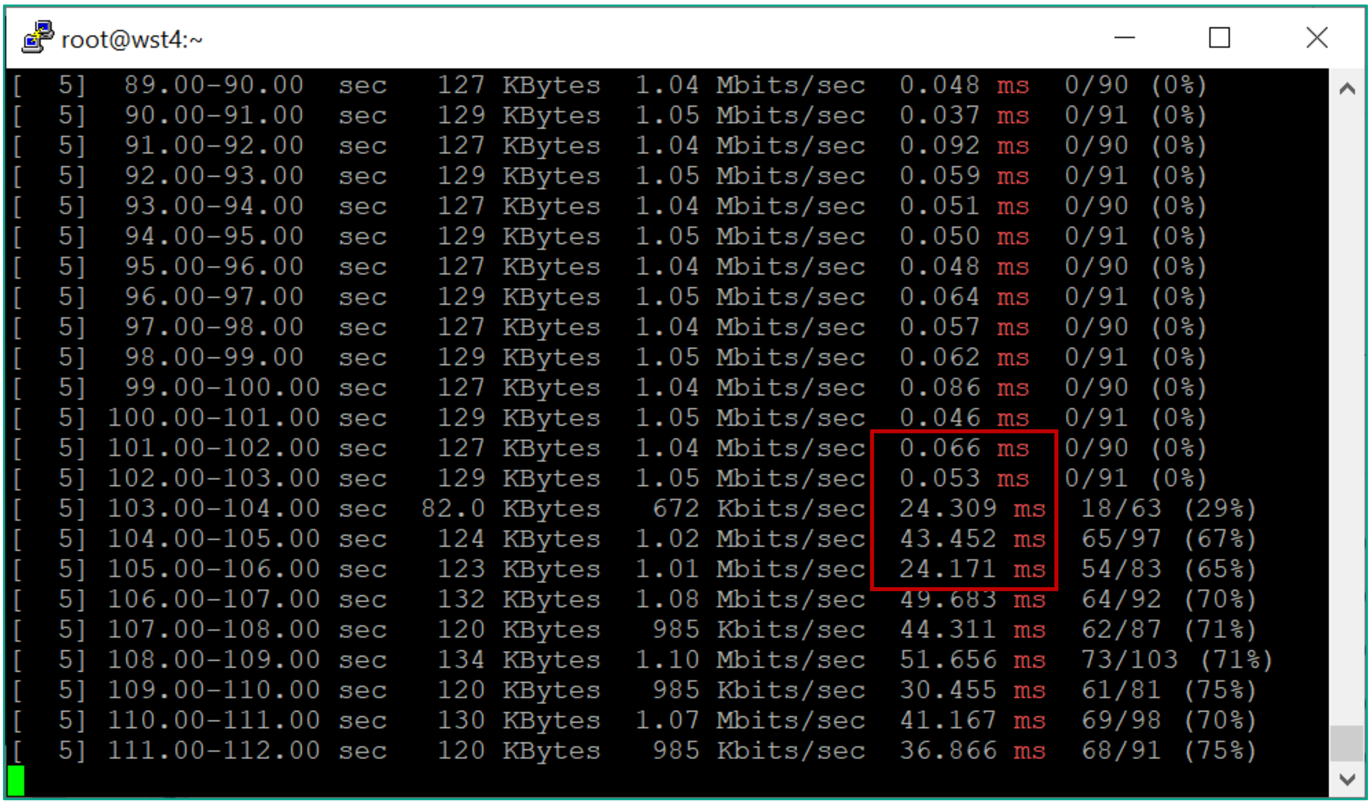
This command creates a **delay** of **300ms** with a **jitter** of **100ms**.

Verify the applied settings with the following command:

```
tc qdisc show
```



Check **jitter** in the **iperf** statistics on the **wst4** host.



Note: default balancing mode is per-flow, so if a traffic passes over a different interface, jitter will not be reflected in the iperf statistics.

If the jitter is not seen in the statistics, then you need to apply emulation on a different interface on the isp host (towards **sdwan0** **(eth0)** interface of the **vCPE-3**):

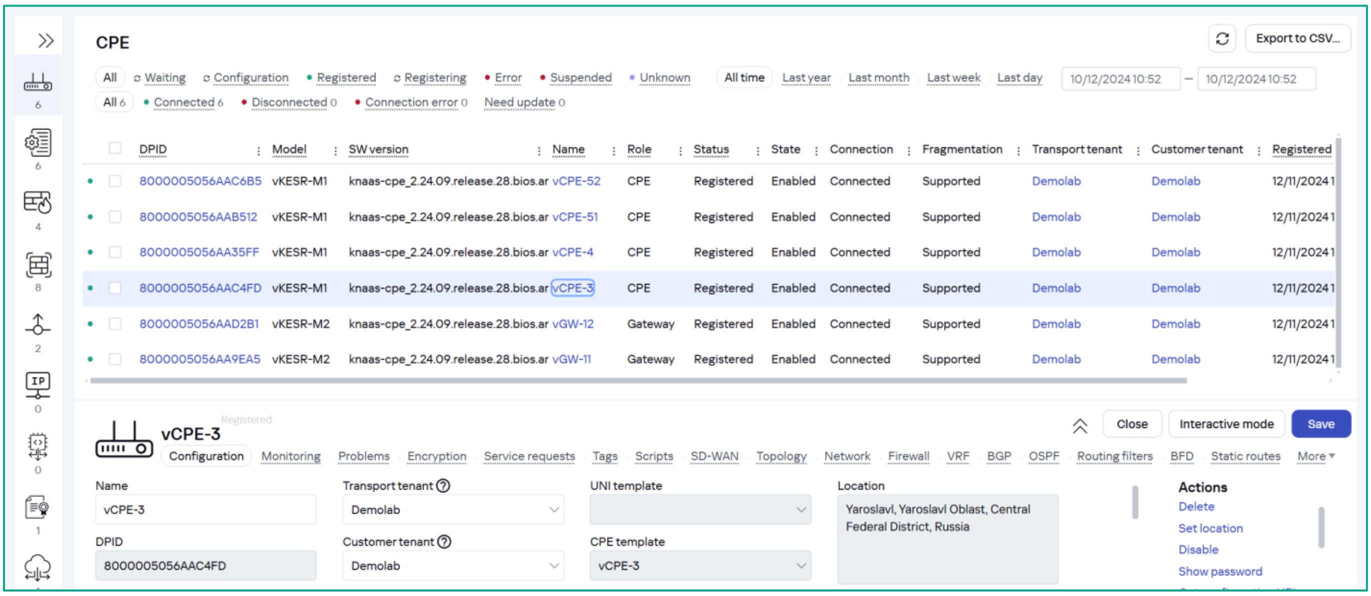
```
tc qdisc add dev ens161 root netem delay 300ms 100ms
```

Remove the delay from the first network interface (towards **sdwan1** **(eth1)** interface of the **vCPE-3**):

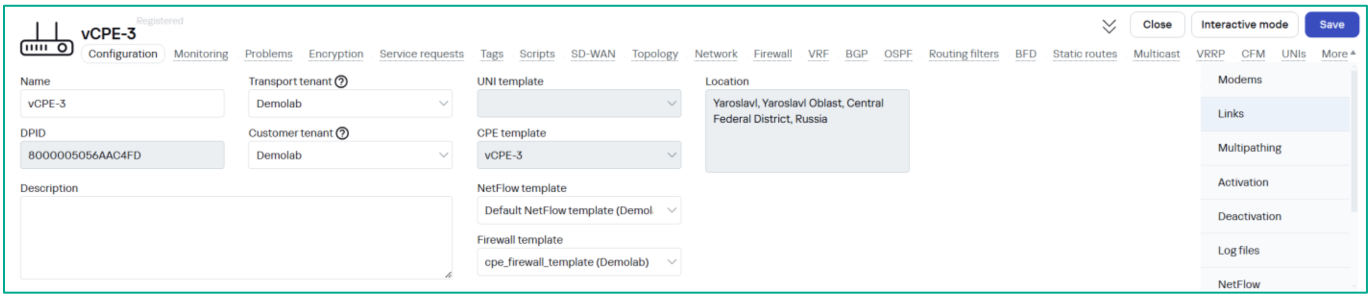
```
tc qdisc del dev ens193 root
```

3.5.3. Enable latency monitoring on vCPE-3 links.

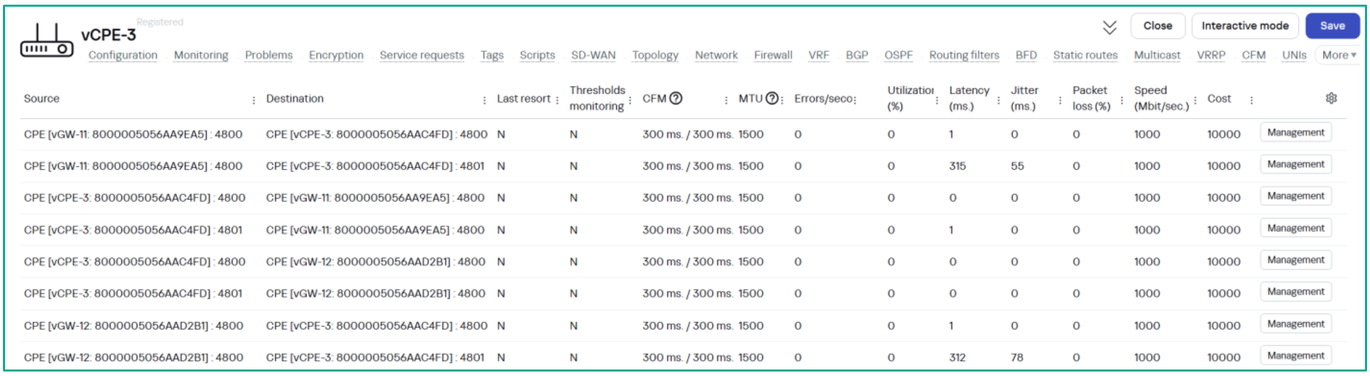
Go to the **CPE** menu and select **vCPE-3**.



Switch to the **Links** tab.



A list of links established with **vCPE-3** is displayed.

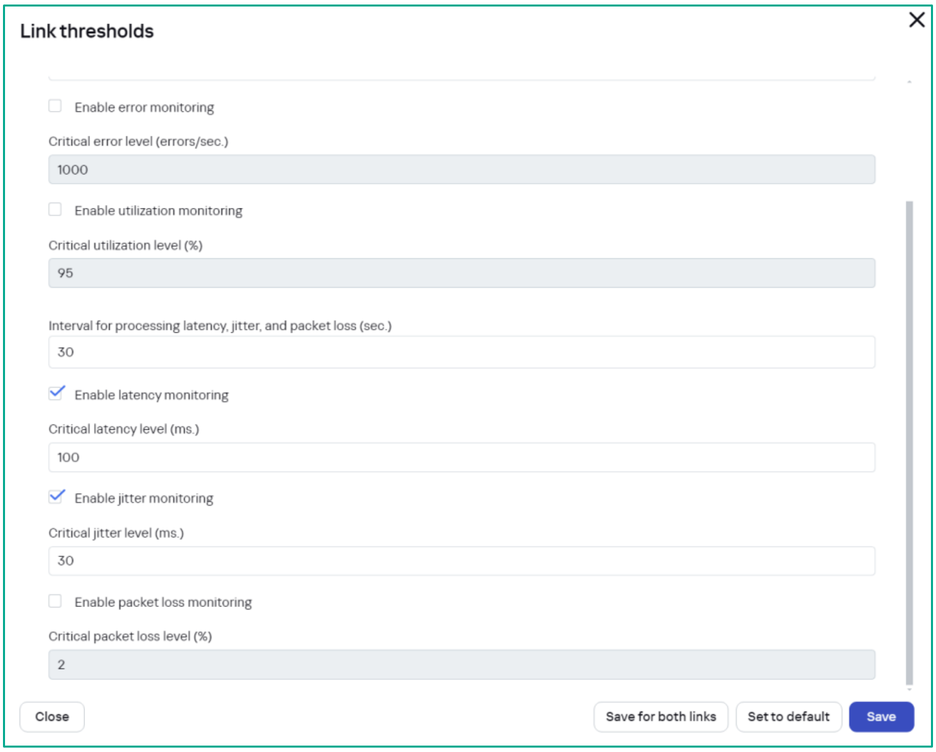


Click **Management → Set thresholds** for all links.

Set link monitoring settings:

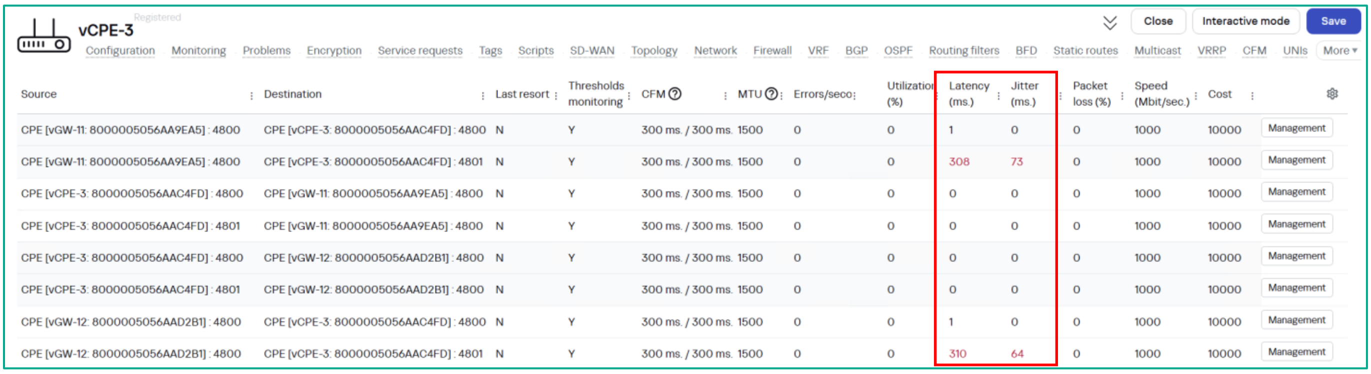
- Check **Enable tunnel thresholds monitoring**.
- **Enable latency monitoring → Critical latency level**: **100 msec**.
- **Enable jittter monitoring → Critical jitter level**: **30 msec**.

Click **Save for both links**.



These settings will enable latency and jitter monitoring for the links and set the thresholds to 100ms and 30ms respectively.

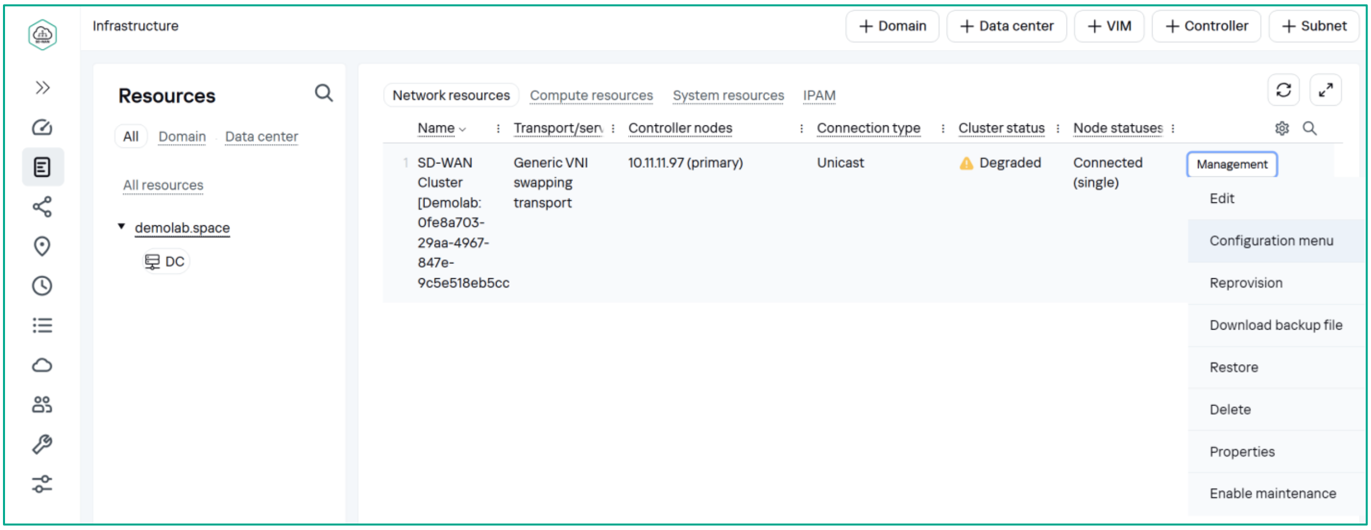
After applying the monitoring settings, the statistics of delays and jitter on links will be displayed. Values of measured parameters that do not meet the thresholds will be highlighted in red color.



Source	Destination	Last resort	Thresholds monitoring	CFM	MTU	Errors/seco	Utilization (%)	Latency (ms.)	Jitter (ms.)	Packet loss (%)	Speed (Mbit/sec.)	Cost	
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	Y	300 ms. / 300 ms.	1500	0	0	308	73	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	Y	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	Y	300 ms. / 300 ms.	1500	0	0	310	64	0	1000	10000	Management

3.5.4. Create a constraint to exclude links that do not meet the specified delay and jitter thresholds. Constraints must be created for redirecting traffic.

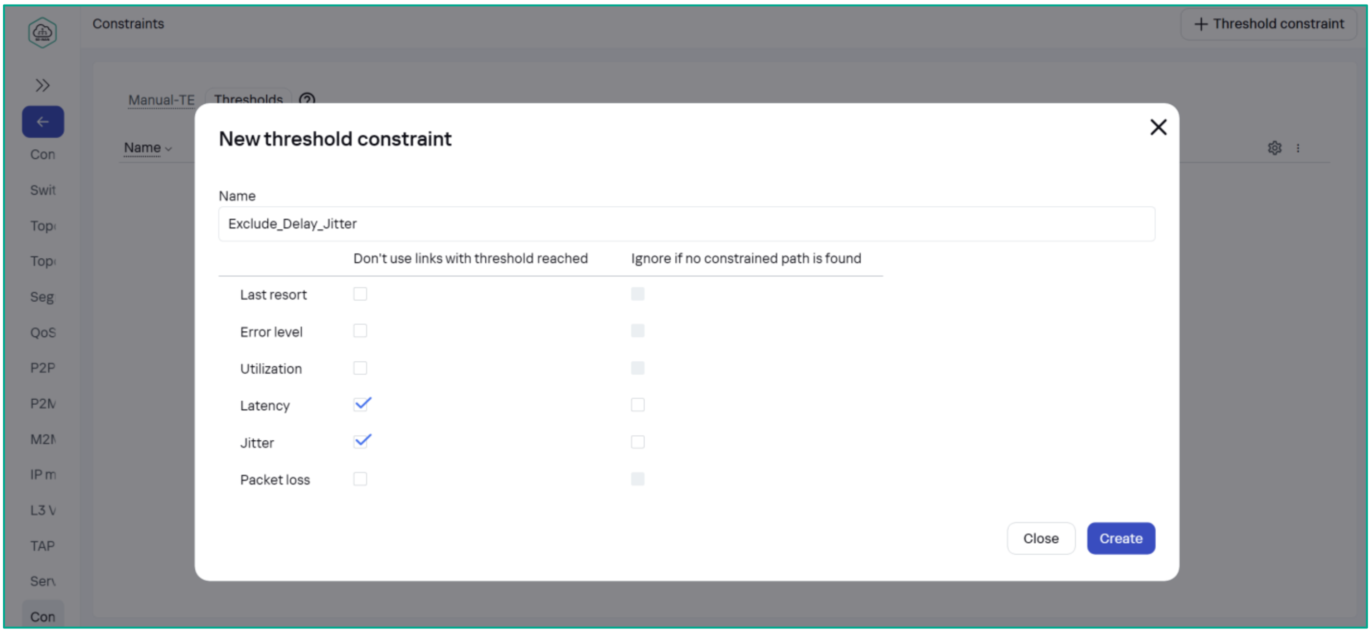
Go to **Infrastructure → SD-WAN cluster → Configuration menu**.



Go to the **Constraints** menu, then switch to the **Thresholds** tab and click **+ Threshold Constraint**.

Set the new constraint settings:

- **Name**: **Exclude_Delay_Jitter**.
- Check **Latency**.
- Check **Jitter**.



Click **Create**.

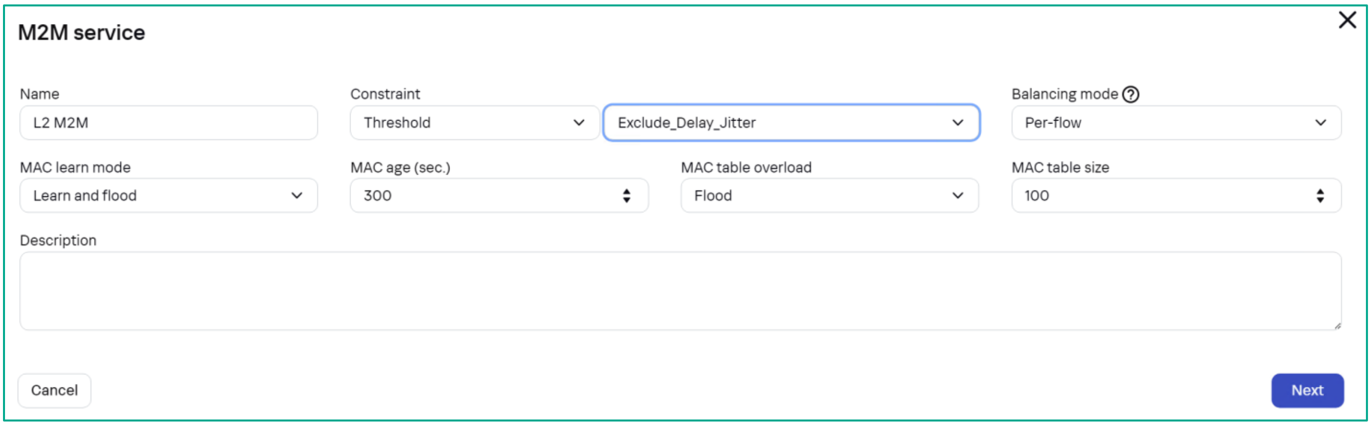
This constraint will exclude links that do not meet the thresholds configured in 3.5.3 from the traffic paths.

3.5.5. Apply constraint to the traffic service.

Go to **M2M Services** menu.

Select **L2 M2M** for editing: **Management → Edit**.

Select the **Threshold** created in step 3.5.4 in the **Constraint** section**.**

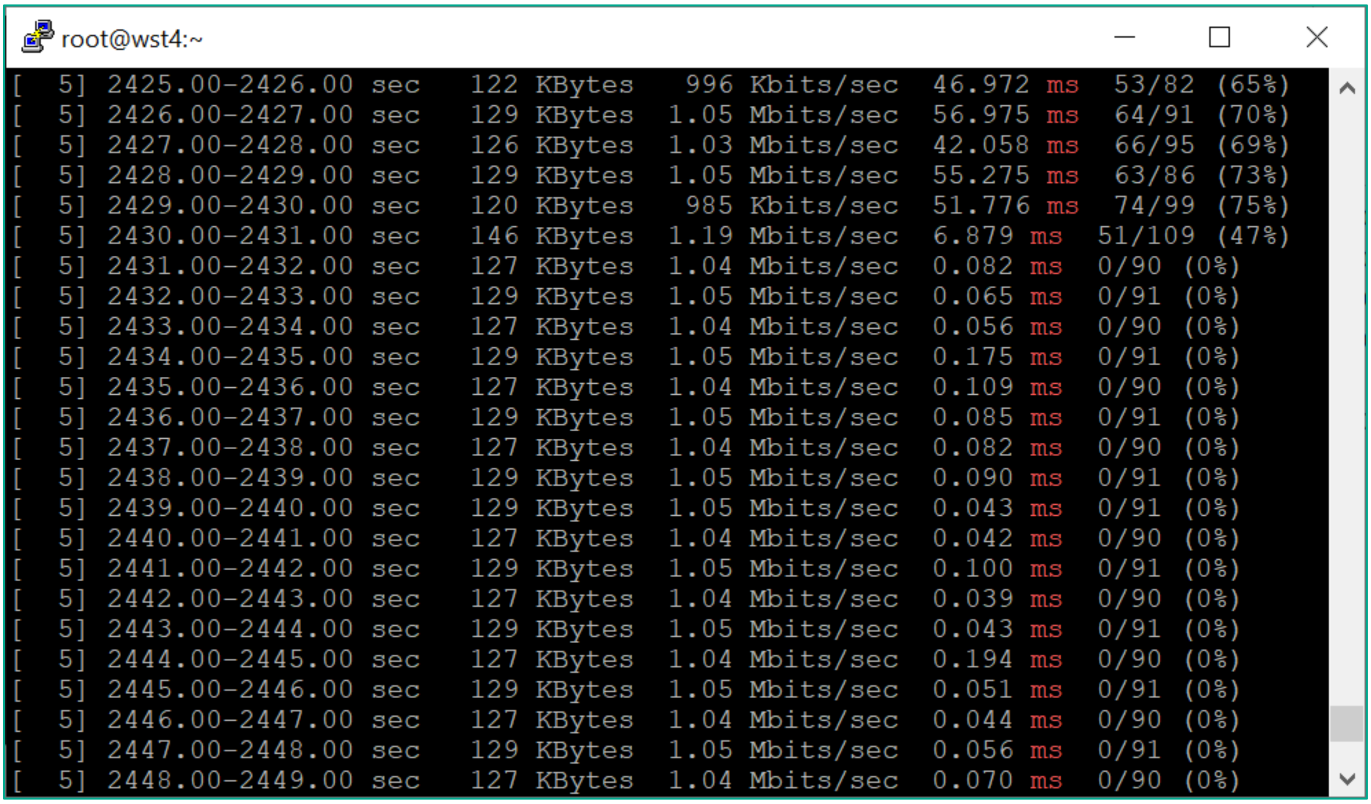


Click **Next**, **Next** and **Save**.

After applying the constraint, the SD-WAN controller will exclude traffic from links that do not meet the constraint applied to the transport service.

3.5.6. Verify traffic redirection.

The **iperf** statistics on **wst4** indicate the absence of jitter because the SD-WAN controller excluded links passing through the first WAN interface of **vCPE-3**, to which latency and jitter emulation was applied.



```
root@wst4:~
[  5] 2425.00-2426.00 sec   122 KBytes   996 Kbits/sec  46.972 ms  53/82 (65%)
[  5] 2426.00-2427.00 sec   129 KBytes  1.05 Mbits/sec  56.975 ms  64/91 (70%)
[  5] 2427.00-2428.00 sec   126 KBytes  1.03 Mbits/sec  42.058 ms  66/95 (69%)
[  5] 2428.00-2429.00 sec   129 KBytes  1.05 Mbits/sec  55.275 ms  63/86 (73%)
[  5] 2429.00-2430.00 sec   120 KBytes   985 Kbits/sec  51.776 ms  74/99 (75%)
[  5] 2430.00-2431.00 sec   146 KBytes  1.19 Mbits/sec  6.879 ms  51/109 (47%)
[  5] 2431.00-2432.00 sec   127 KBytes  1.04 Mbits/sec  0.082 ms  0/90 (0%)
[  5] 2432.00-2433.00 sec   129 KBytes  1.05 Mbits/sec  0.065 ms  0/91 (0%)
[  5] 2433.00-2434.00 sec   127 KBytes  1.04 Mbits/sec  0.056 ms  0/90 (0%)
[  5] 2434.00-2435.00 sec   129 KBytes  1.05 Mbits/sec  0.175 ms  0/91 (0%)
[  5] 2435.00-2436.00 sec   127 KBytes  1.04 Mbits/sec  0.109 ms  0/90 (0%)
[  5] 2436.00-2437.00 sec   129 KBytes  1.05 Mbits/sec  0.085 ms  0/91 (0%)
[  5] 2437.00-2438.00 sec   127 KBytes  1.04 Mbits/sec  0.082 ms  0/90 (0%)
[  5] 2438.00-2439.00 sec   129 KBytes  1.05 Mbits/sec  0.090 ms  0/91 (0%)
[  5] 2439.00-2440.00 sec   129 KBytes  1.05 Mbits/sec  0.043 ms  0/91 (0%)
[  5] 2440.00-2441.00 sec   127 KBytes  1.04 Mbits/sec  0.042 ms  0/90 (0%)
[  5] 2441.00-2442.00 sec   129 KBytes  1.05 Mbits/sec  0.100 ms  0/91 (0%)
[  5] 2442.00-2443.00 sec   127 KBytes  1.04 Mbits/sec  0.039 ms  0/90 (0%)
[  5] 2443.00-2444.00 sec   129 KBytes  1.05 Mbits/sec  0.043 ms  0/91 (0%)
[  5] 2444.00-2445.00 sec   127 KBytes  1.04 Mbits/sec  0.194 ms  0/90 (0%)
[  5] 2445.00-2446.00 sec   129 KBytes  1.05 Mbits/sec  0.051 ms  0/91 (0%)
[  5] 2446.00-2447.00 sec   127 KBytes  1.04 Mbits/sec  0.044 ms  0/90 (0%)
[  5] 2447.00-2448.00 sec   129 KBytes  1.05 Mbits/sec  0.056 ms  0/91 (0%)
[  5] 2448.00-2449.00 sec   127 KBytes  1.04 Mbits/sec  0.070 ms  0/90 (0%)
```

3.5.7. Restore the settings after the test is completed.

Remove the **constraint** from the transport service: repeat step 3.5.5.

Disable jitter and latency emulation on the **isp** host.

```
tc qdisc del dev ens161 root

tc qdisc del dev ens193 root
```

Disable **latency** and **jitter** monitoring on **vCPE-3** links: repeat step 3.4.3.

Stop **iperf** on **wst3** and **wst4**, started in 3.5.3 (you can stop it with `Ctrl+Z`).

3.6. **Traffic prioritization with ACLs**

The SD-WAN solution allows you to create traffic classifiers based on IP/TCP/UDP header fields and direct traffic to specific transport services. For example, it is possible to create a prioritized service for delay-sensitive traffic with restrictions so that traffic does not pass through links with a delay that does not meet a specified constraint.

For more information, please refer to Kaspersky SD-WAN Online Help: https://support.kaspersky.com/help/SD-WAN/2.3/en-US/246544.htm

In this scenario, a UDP port-based traffic classifier is created to redirect test traffic to the prioritized service.

Test traffic will run between the wst3 and wst4 hosts using iperf3 on UDP port 5555. An L3 ACL will be created to categorize the test traffic and an ACL interface will be created to redirect the traffic to a specific service.

Links passing through the sdwan0 (eth0) vCPE-3 interface will be labeled as Last resort and a new transport service will be created. Constraints will be set for the new service to exclude links marked as Last resort from the traffic path. The tcpdump tool on vCPE-3 will be used to verify traffic path.

3.6.1. Generate test traffic between wst3 and wst4.

Start the **iperf3** server on the **wst4** host with port **5555**:

```
iperf3 -s -p 5555
```

Start the **iperf3** client on the **wst3** host with **UDP** port **5555**:

```
iperf3 -u -t 6000 -c <wst4 IP address> -p 5555
```

```
root@wst3:~

[root@wst3 ~]# iperf3 -u -t 6000 -c 10.20.4.223 -p 5555
Connecting to host 10.20.4.223, port 5555
[  4] local 10.20.3.188 port 51821 connected to 10.20.4.223 port 5555
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   116 KBytes   950 Kbits/sec  82
[  4]   1.00-2.00   sec   129 KBytes  1.05 Mbits/sec  91
[  4]   2.00-3.00   sec   127 KBytes  1.04 Mbits/sec  90
[  4]   3.00-4.00   sec   129 KBytes  1.05 Mbits/sec  91
[  4]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec  90
```

3.6.2. Identify the tunnel interface through which the test traffic flows.

Connect to **vCPE-3** via the SSH and start **tcpdump** to check through which interface the traffic flows: **genev_sys_4800** or **genev_sys_4801**:

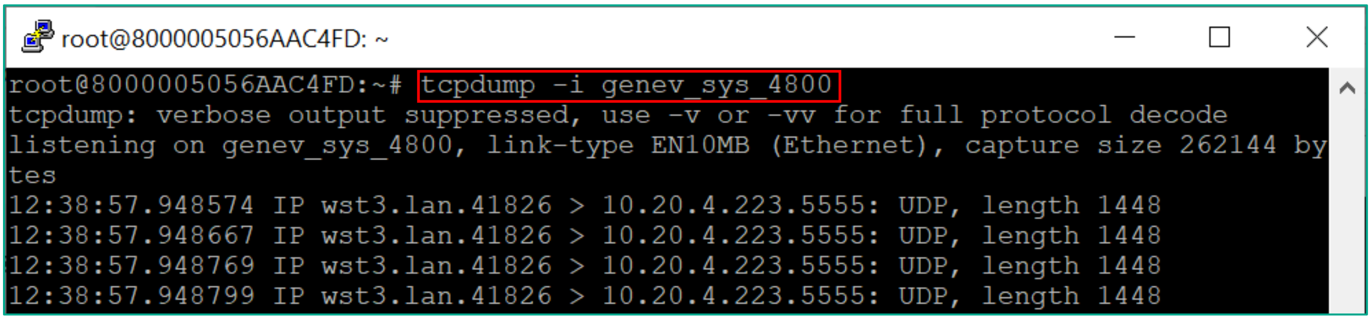
```
tcpdump -i genev_sys_4800
```

If the test traffic is going through the tunnel interface, the tcpdump output will show UDP packets sent by iperf3 on port 5555.

From the **tcpdump** output, determine which interface the test traffic is passing through: **genev_sys_4800** or **genev_sys_4801**.

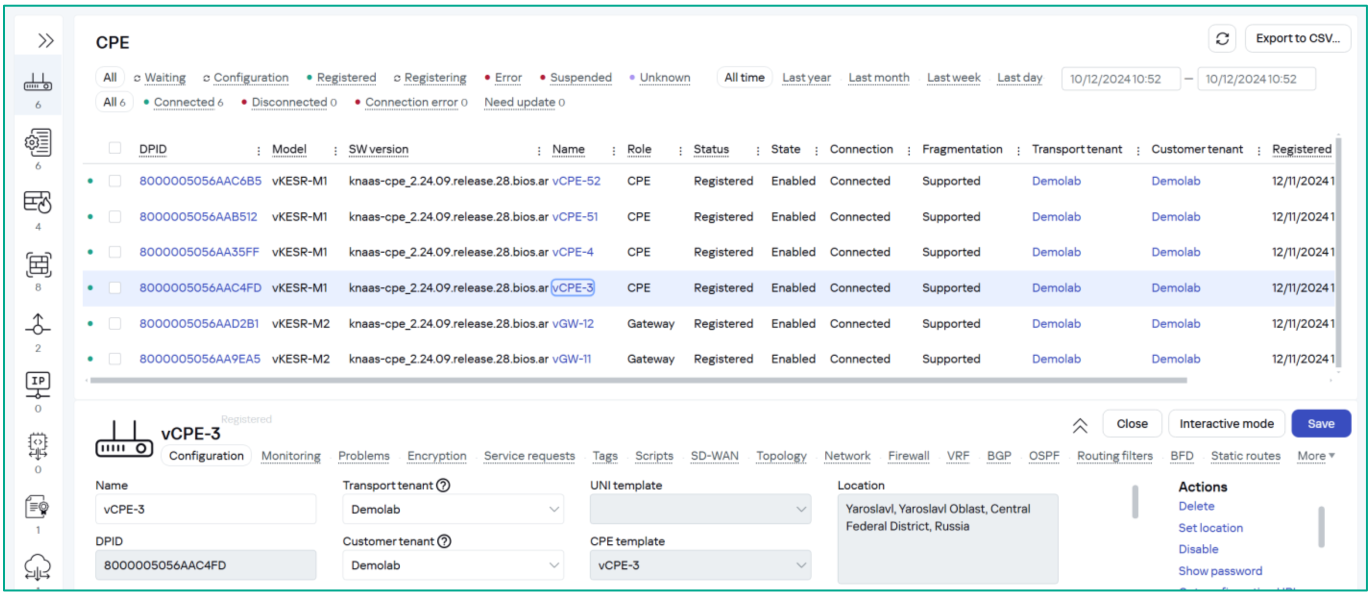
genev_sys_4800 and **4801** are the CPE tunnel Interfaces. Each port number corresponds to a WAN interface number. The numbers are assigned consecutively, starting with port 4800, one for each WAN interface. Port **4800** is designated for WAN interface **sdwan0** (**eth0**), while port **4801** corresponds to WAN interface **sdwan1** (**eth1**).

In this example, the traffic flows through the **genev_sys_4800** interface.

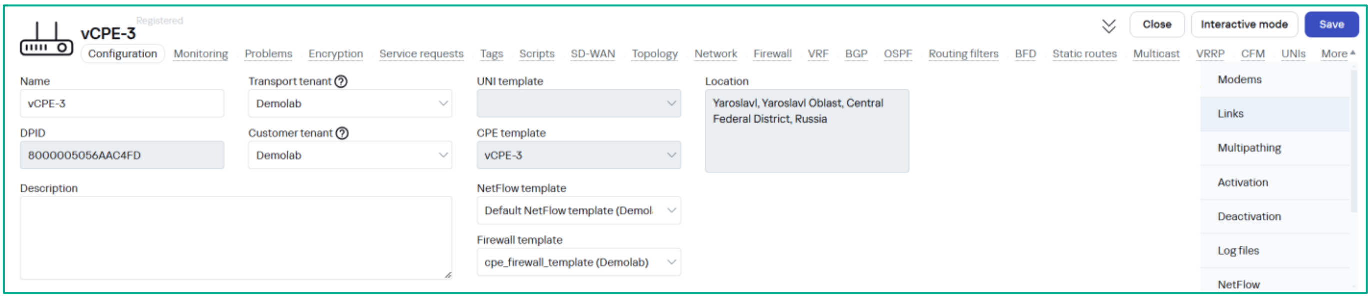


3.6.3. Set the Last resort parameter for vCPE-3 links.

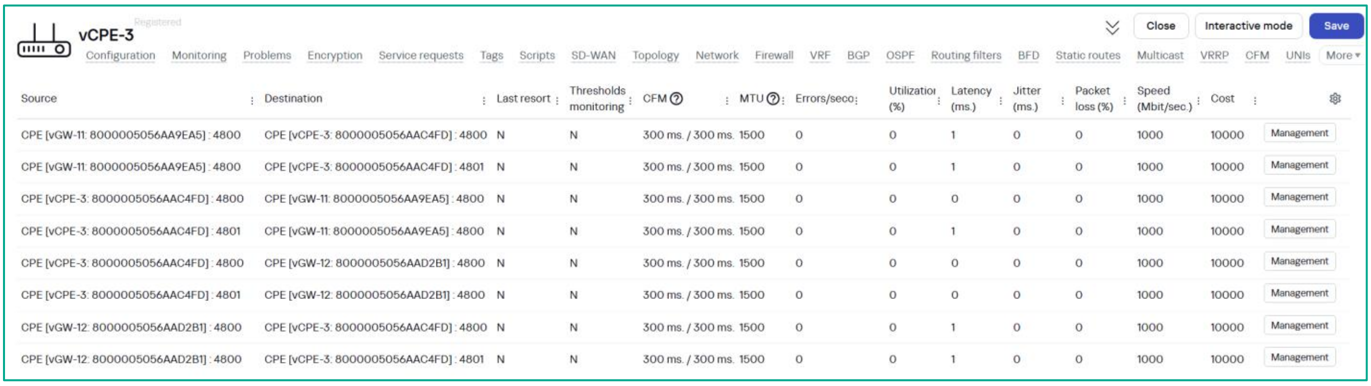
Go to the **CPE** menu and select **vCPE-3**.



Switch to the **Links** tab.



A list of links established with **vCPE-3** is displayed.



Source	Destination	Last resort	Thresholds monitoring	CFM	MTU	Errors/seco	Utilization (%)	Latency (ms.)	Jitter (ms.)	Packet loss (%)	Speed (Mbit/sec.)	Cost	
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management

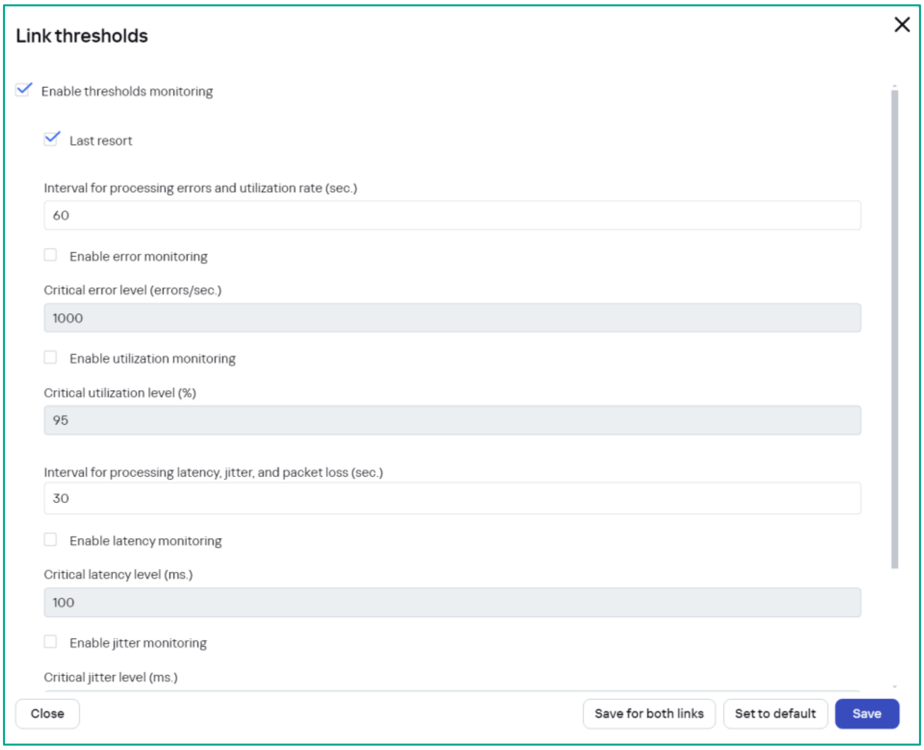
Find all links through which traffic flows: the source and destination ports of the links (4800 or 4801) must match the interface number according to the check in 3.6.2. In this example, the result of the check is that traffic flows through the **genev_sys_4800** link.

The links through which traffic flows in this example:

- **vCPE-3**:**4800** - vGW-11:4800
- **vCPE-3**:**4800** - vGW-12:4800
- vGW-11:4800 - **vCPE-3**:**4800**
- vGW-12:4800 - **vCPE-3**:**4800**

For each link with port 4800 for vCPE-3, click **Management → Set thresholds** to set **Last resort** parameter:

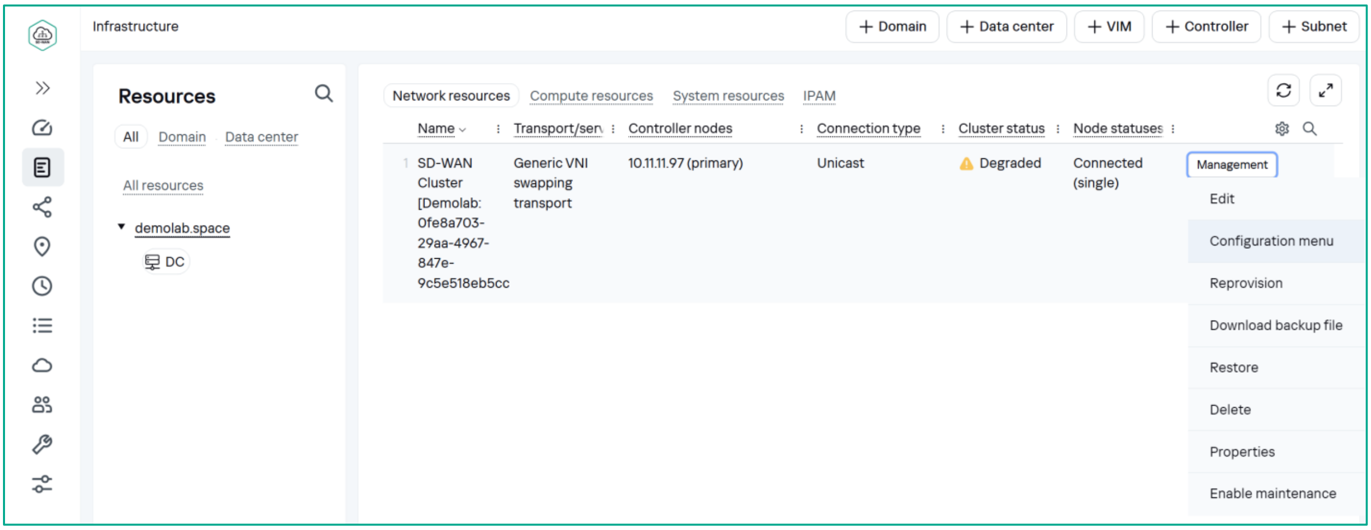
- Check **Enable tunnel thresholds monitoring**.
- Check **Last resort**.



Click **Save for both links**.

3.6.4. Create a constraint to exclude links with Last resort parameter set.

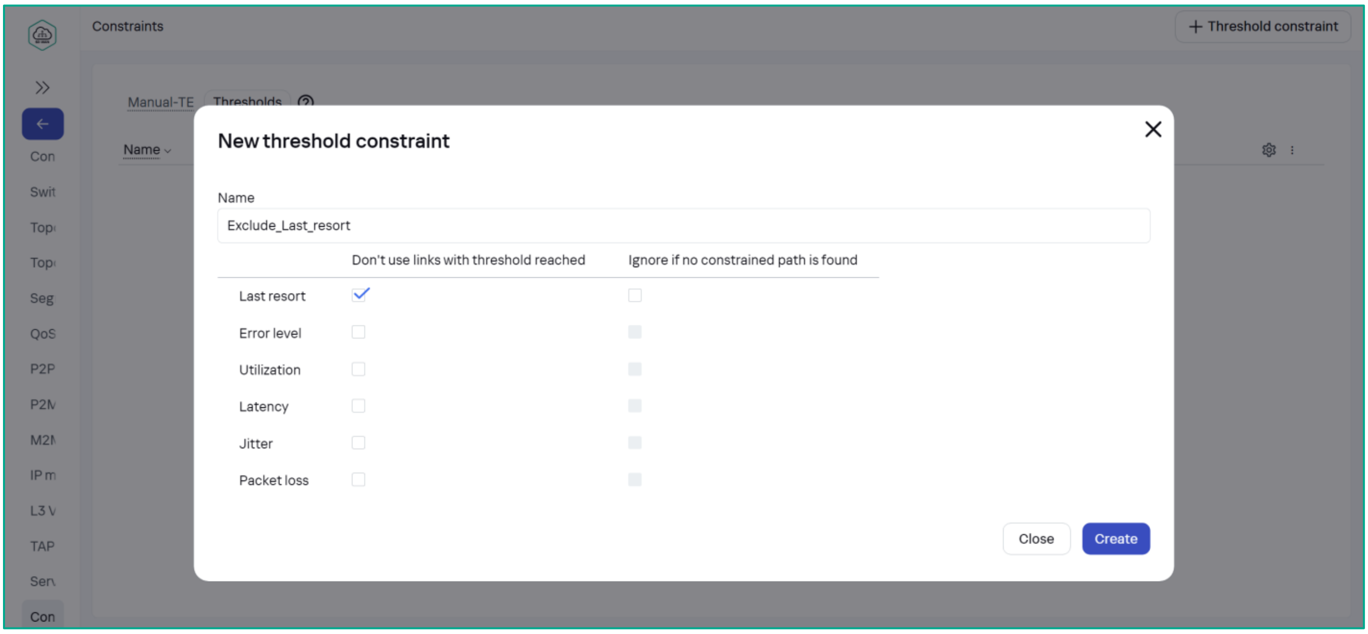
Constraints must be created to redirect traffic. Go to **Infrastructure → SD-WAN cluster → Configuration menu**.



Go to the **Constraints** menu, then open the **Thresholds** tab and click the **+ Threshold Constraint** button.

Set constraint parameters:

- **Name**: **Exclude_Last_resort**.
- Check **Last resort**.



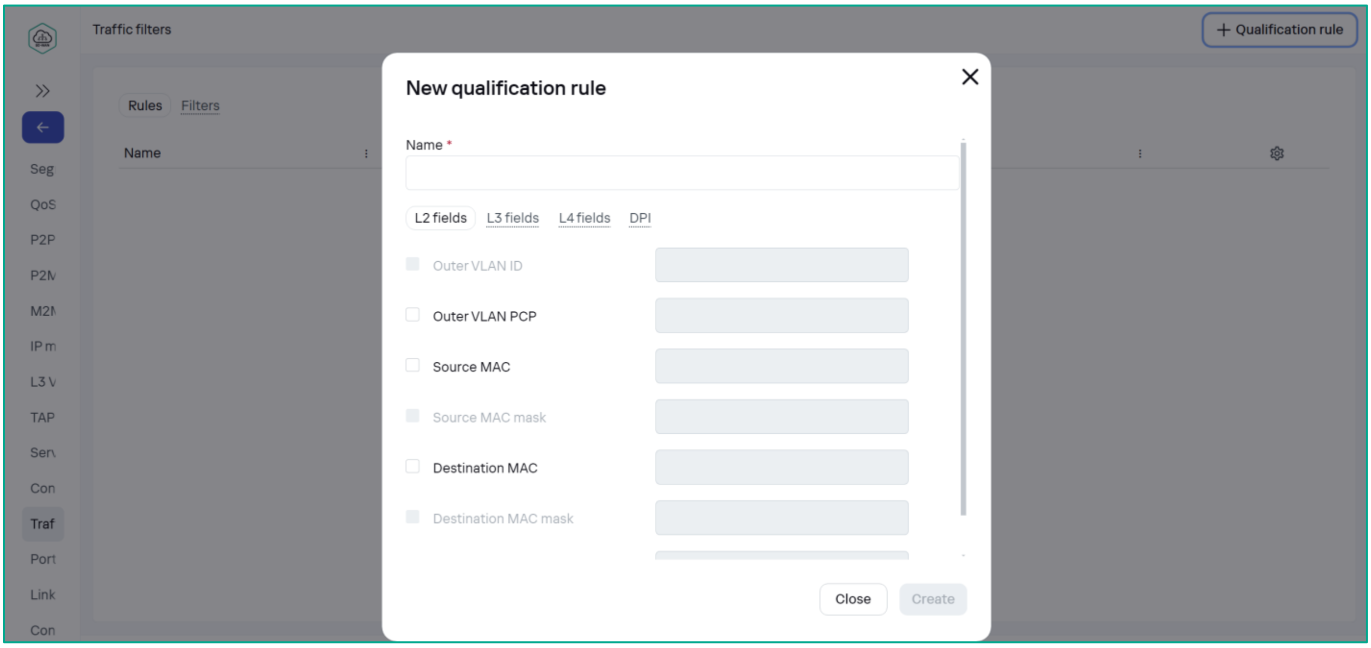
Click **Create**.

This constraint will exclude links marked as Last resort from the transport service.

3.6.5. Create traffic qualification rule.

To redirect traffic to a separate service, traffic filter must be created to capture the test UDP traffic with port 5555.

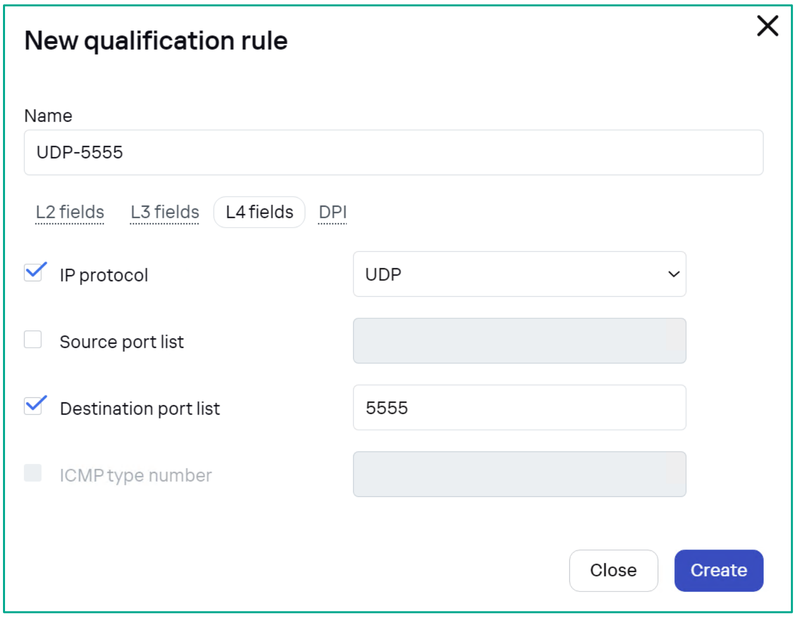
Go to **Traffic Filters**. Then go to the **Rules** tab and click + **Qualification rule**.



Set rule parameters:

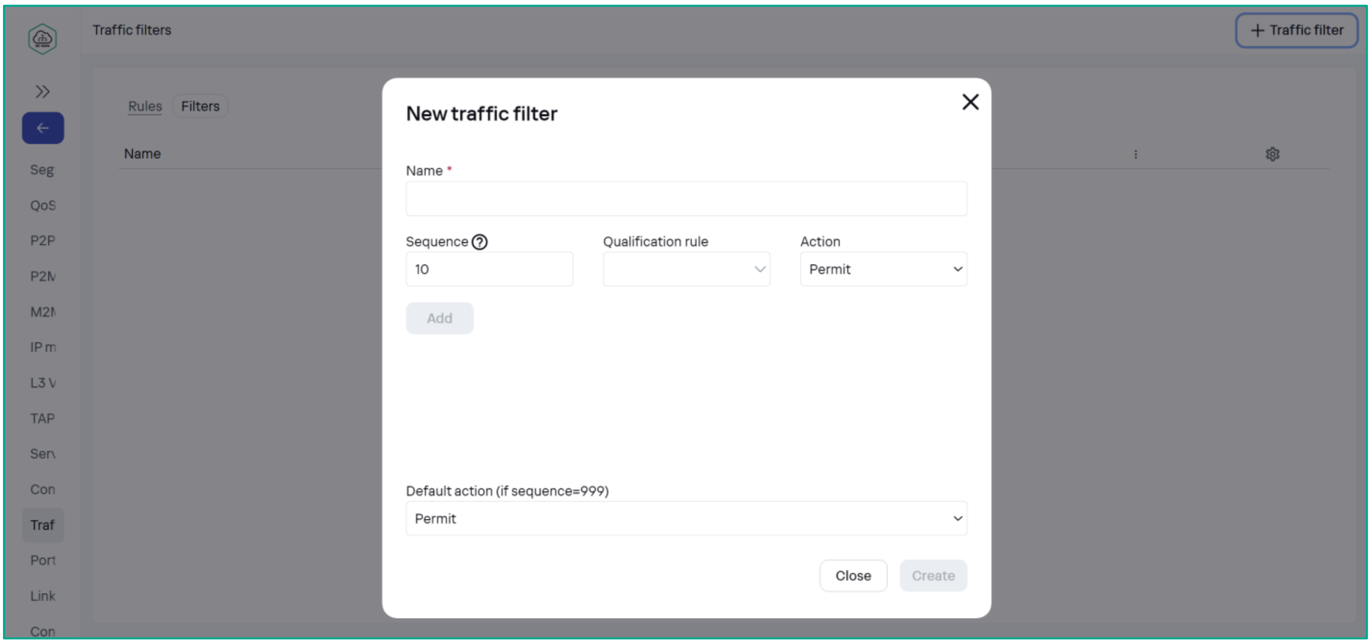
- **Name**: **UDP-5555**
- **L3 Fields**:
    - **Protocol**: **IPv4**
- **L4 Fields**:
    - **IP protocol**: **UDP**
    - **Destination port list**: **5555**

Click **Create**.



3.6.6. Create a traffic filter to redirect test traffic to a separate service.

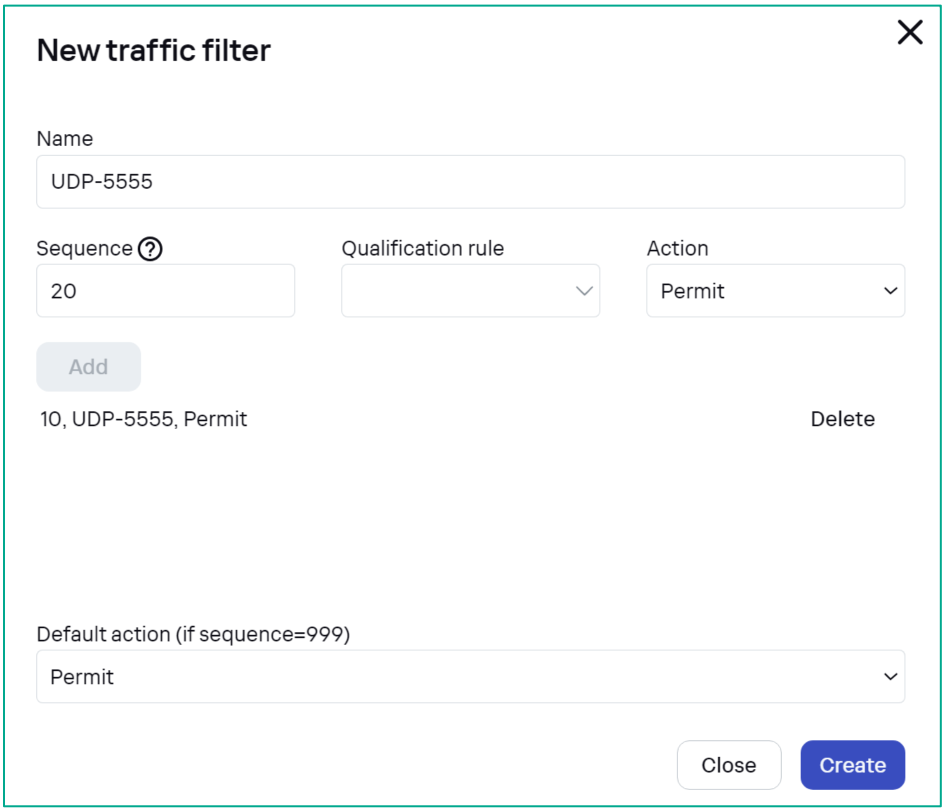
Go to **Traffic filters** tab, click + **Traffic filter**.



Set traffic filter parameters:

- **Name**: **UDP-5555**.
- Add **Qualification rule**: Select rule, created in 3.6.5 (**UDP-5555**), set **Action**: **Permit**. Click **Add**.

Click **Create**.



3.6.7. Create ACL Service interfaces.

Traffic enters the transport service through service interfaces. It is necessary to create a special ACL Service Interface (ACL SI) to direct filtered traffic to the transport service.

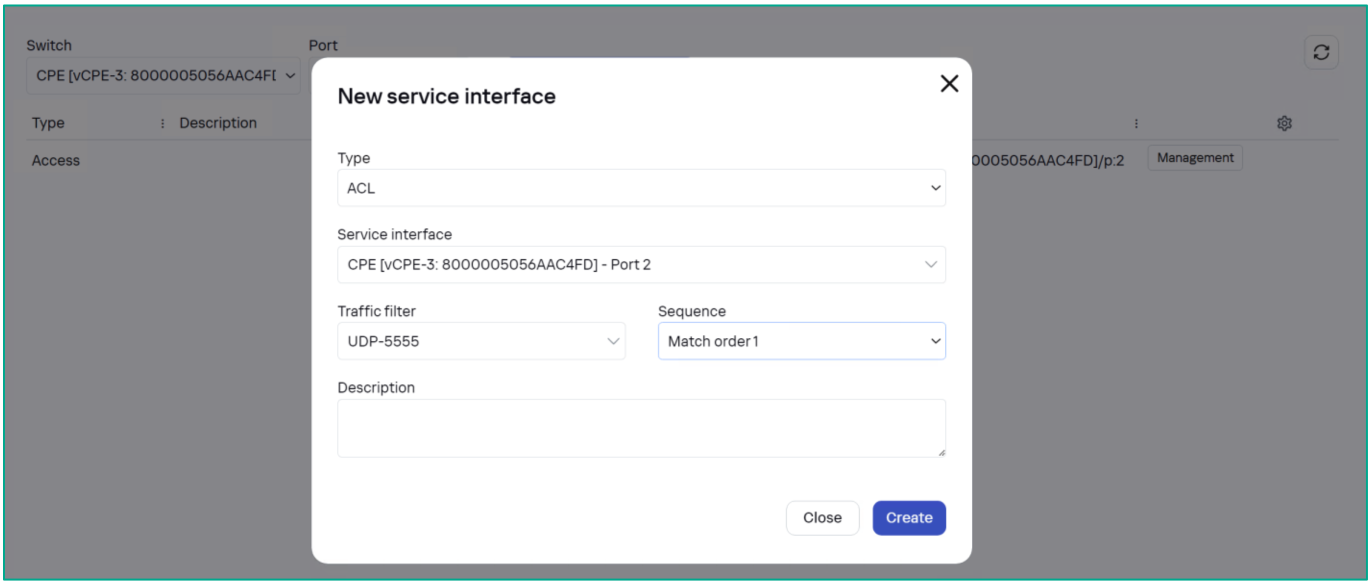
Go to the **Service Interfaces** tab and select **Switch**: **vCPE-3** and **Port**: **2 (ovs-lan).**

Click **Create service interface**.

Set service interface parameters:

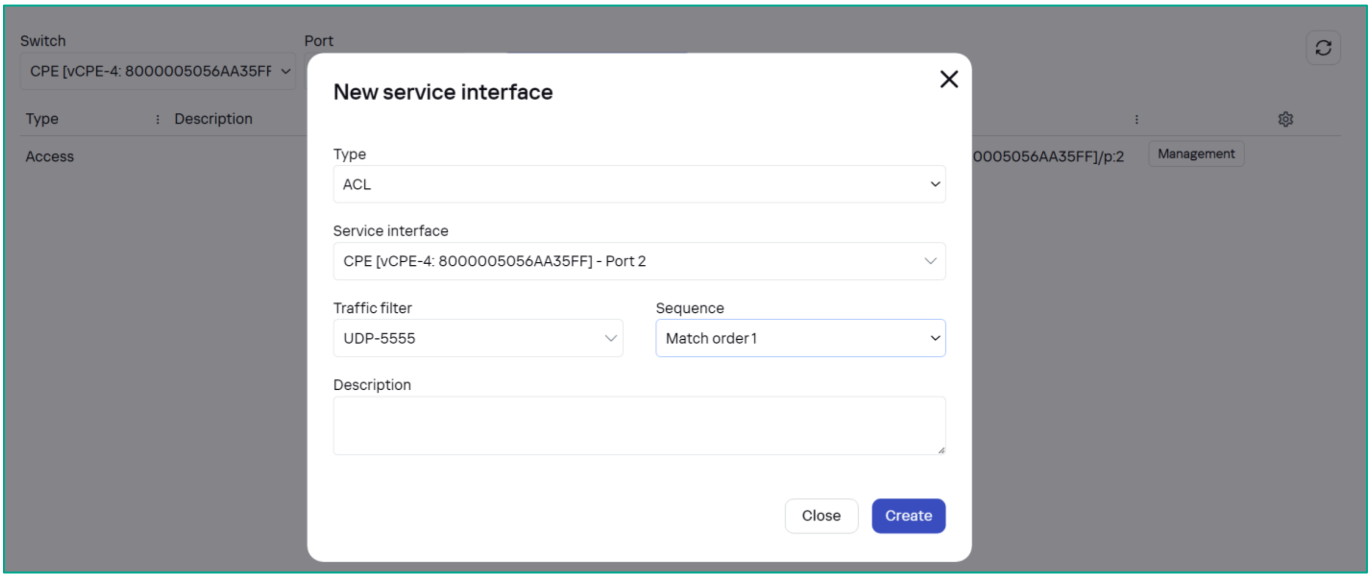
- **Type**: **ACL**.
- **Service interface**: **vCPE-3** - **Port 2**.
- **Traffic Filter** for UDP 5555 created in step 3.6.6 (**UDP-5555**).
- **Sequence**: **Match order 1** (this ACL SI will be the first to process traffic).

Click **Create**.



To create a new transport service, you must create service interfaces for each CPE.

Create a similar ACL service interface for **vCPE-4**.



3.6.8. Create a separate transport service for priority traffic.

Go to the **M2M Services** menu, click **+ M2M service**

Set service parameters:

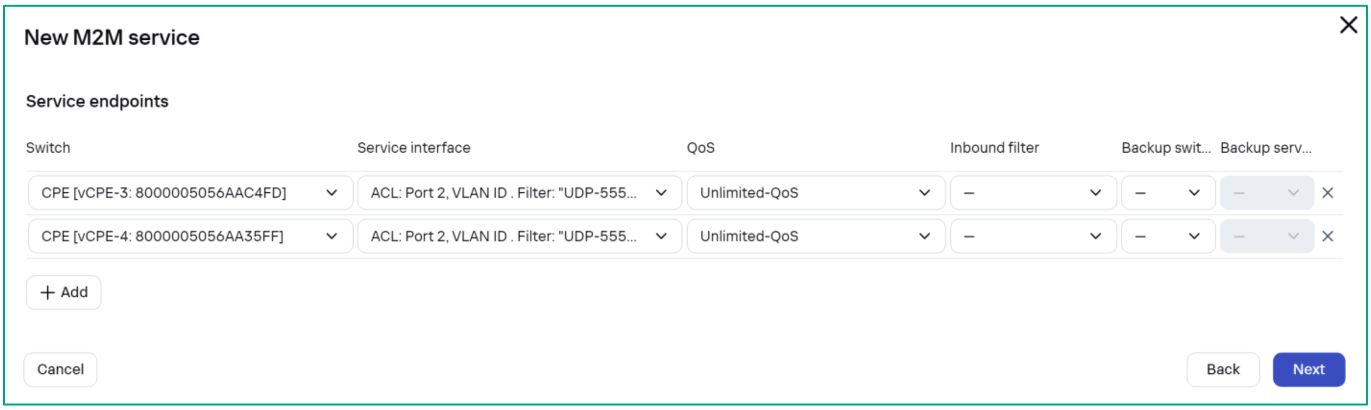
- **Name**: **M2M_ACL**.
- **Constraint**: **Threshold** created in 3.6.4 (**Exclude_Last_resort**).

Click **Next**.

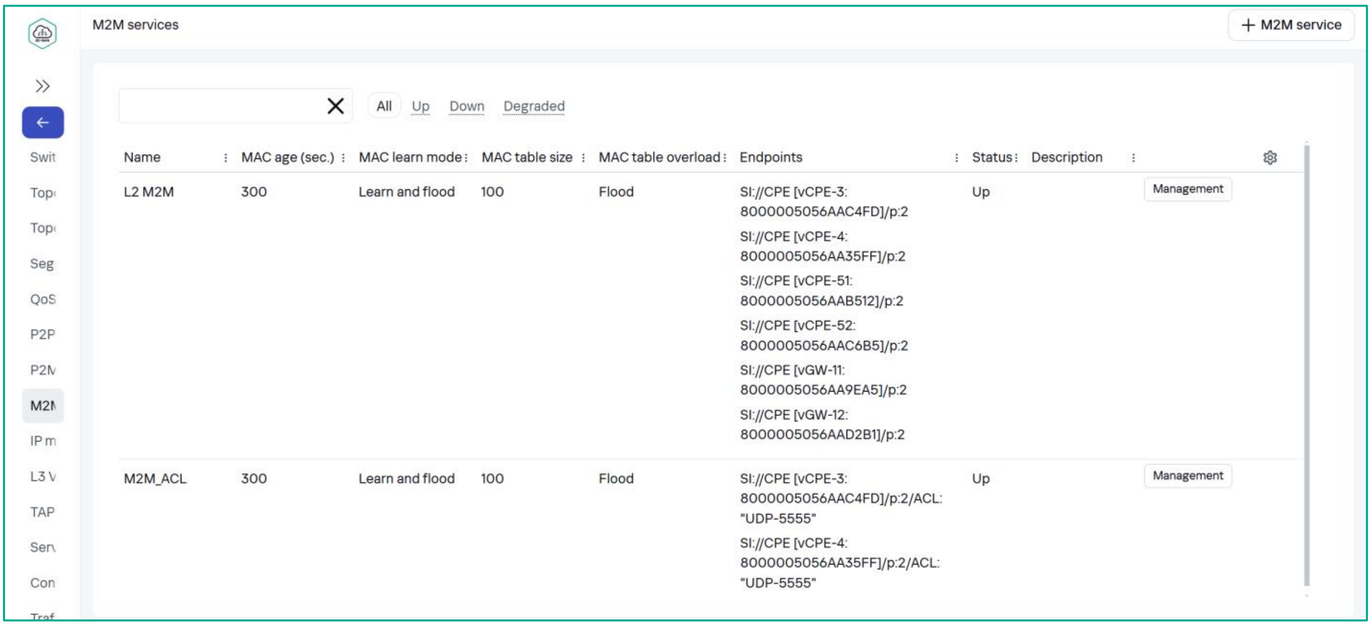
Click **+ Add** in **Service endpoints** section to add service interfaces created in 3.6.7.

Set **service endpoints** parameters:

- **Switch**: **vCPE-3** and **vCPE-4**.
- **Service interface**: created in 3.6.7 **ACL Service Interfaces** for vCPE-3 and vCPE-4.
- **QoS**: **Unlimited QoS**.



Click **Next** and **Create**.



3.6.9. Verify traffic redirection.

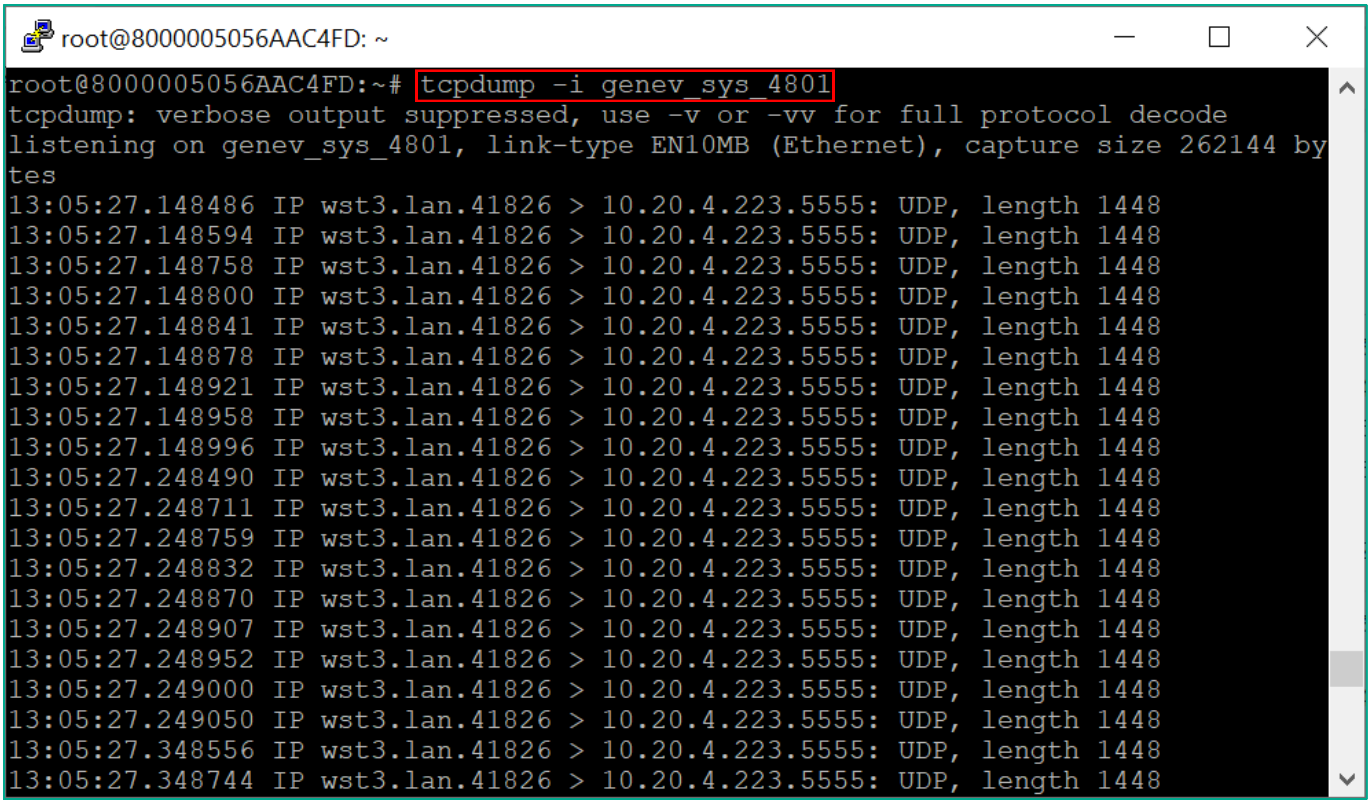
Connect to **vCPE-3** and verify that the traffic is redirected to a second WAN interface (depending on the settings made earlier):

In step 3.6.2 it was verified that the traffic goes through the interface **genev_sys_4800** (**sdwan0**). After configuring a separate transport service with constraints and traffic filtering, the traffic was redirected to interface **genev_sys_4801** (**sdwan1**).

Use **tcpdump** to check traffic on the **genev_sys_4801** interface:

```
tcpdump -i genev_sys_4801
```

The screenshot shows that traffic has switched from interface **genev_sys_4800** (sdwan0) to **genev_sys_4801** (sdwan1).



3.6.10. Restore the settings after the test is completed.

Delete the transport service created in step 3.6.8 (when deleting, check **Delete associated service interfaces**).

Remove the **Last resort** parameter added in 3.6.3 from **vCPE-3** links.

Stop **iperf** on **wst3** and **wst4**, started in step. 3.6.1.

3.7. **Traffic prioritization with DPI**

With the SD-WAN solution, you can use DPI to create traffic classifiers and redirect traffic for specific applications.

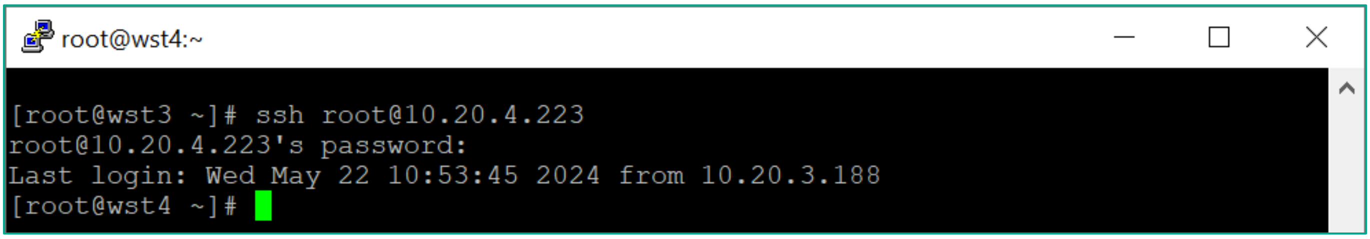
For more information, please refer to Kaspersky SD-WAN Online Help: https://support.kaspersky.com/help/SD-WAN/2.3/en-US/246544.htm

In this scenario, a classifier is created to redirect SSH and HTTP traffic to a prioritized service. Test traffic is generated between workstations wst3 and wst4 using ssh, nc, and curl. A DPI rule is created to classify test traffic and an ACL interface is created to redirect the traffic to a separate service. Links passing through the sdwan0 (eth0) vCPE-3 interface are marked as Last resort and a separate transport service is created with constraints set to exclude Last resort links from the traffic path. The tcpdump is used on vCPE-3 to verify the traffic path.

3.7.1. Generate test traffic between wst3 and wst4.

Start the SSH session on host **wst3** to **wst4**:

```
ssh root@<wst4 IP address>
```



3.7.2. Identify the tunnel interface through which the test traffic flows.

Connect to **vCPE-3** via the SSH and start **tcpdump** to check through which interface the traffic flows: **genev_sys_4800** or **genev_sys_4801**:

```
tcpdump -i genev_sys_4800
```

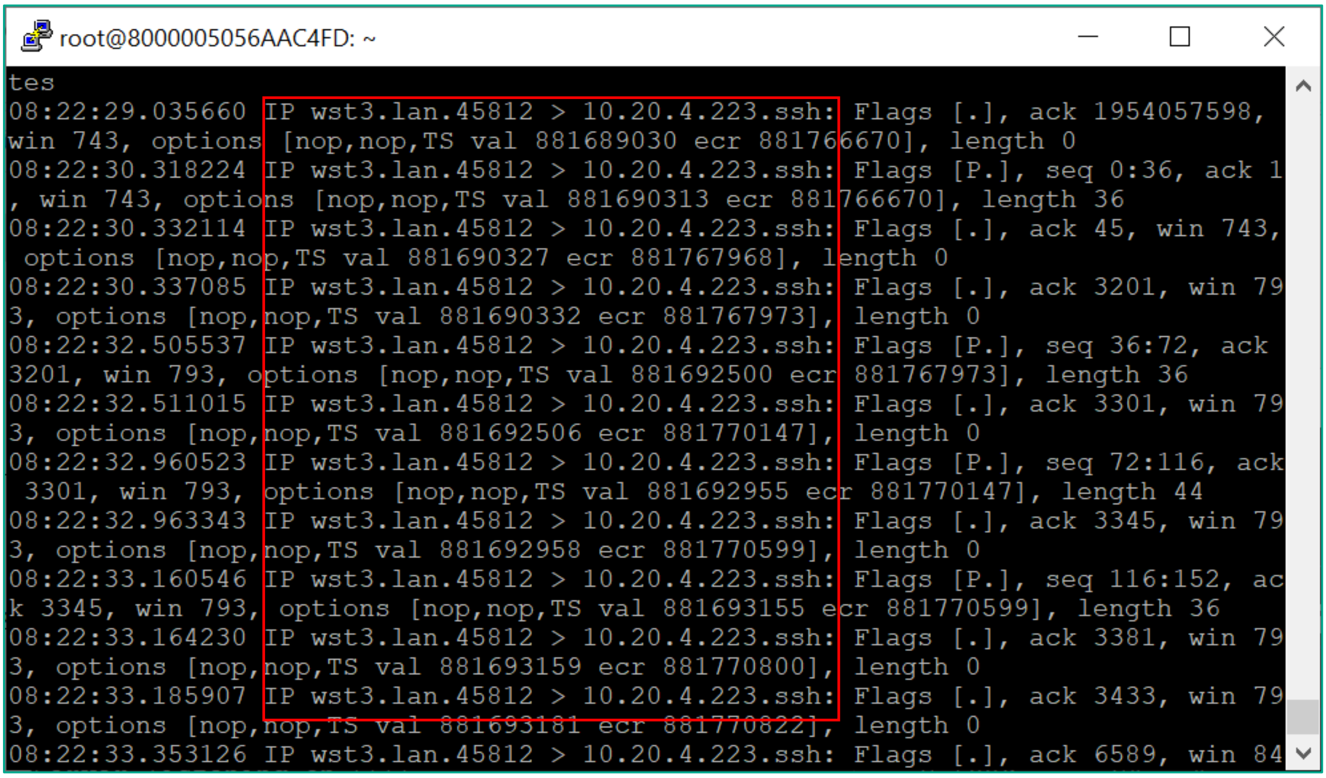
If the test traffic is going through the tunnel interface, the tcpdump output will show packets for SSH session to the wst4.

From the **tcpdump** output, determine which interface the test traffic is passing through: **genev_sys_4800** or **genev_sys_4801**.

genev_sys_4800 and **4801** are the CPE Tunnel Interfaces. Each port number corresponds to a WAN interface number. The numbers are assigned consecutively, starting with port 4800, one for each WAN interface. Port **4800** is designated for WAN interface **sdwan0** (**eth0**), while port **4801** corresponds to WAN interface **sdwan1** (**eth1**).

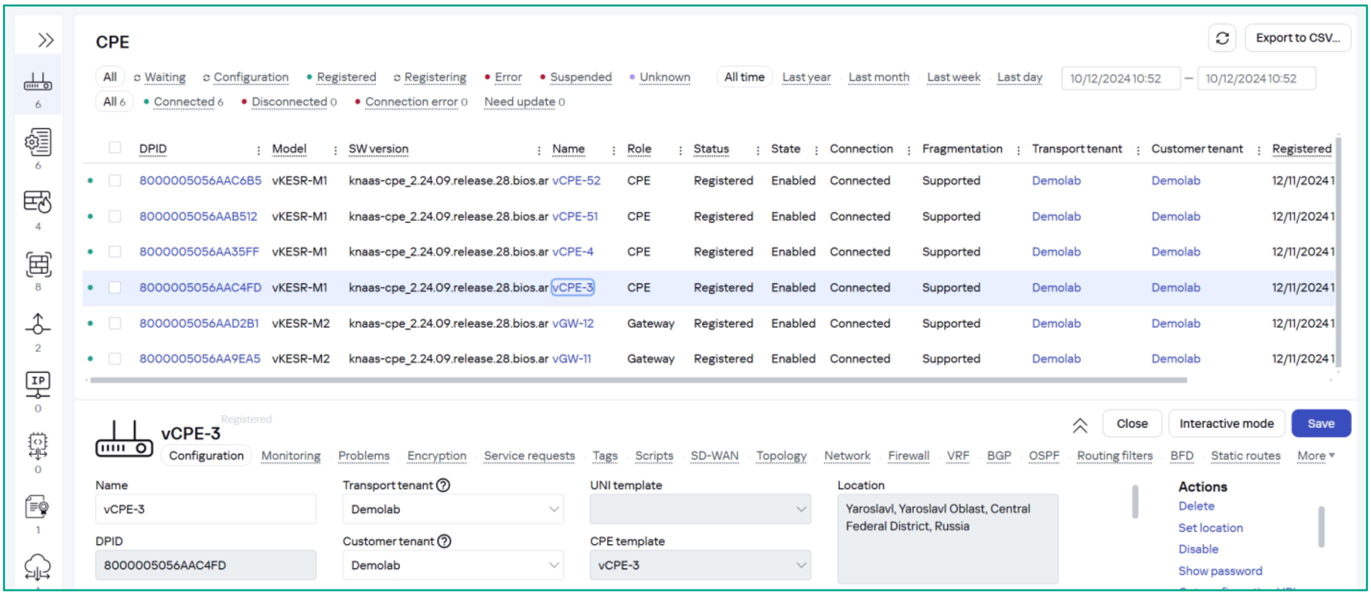
Note which interface is currently carrying the test traffic: for an SSH session, it can be asymmetric (one way through 4800 and the other way through 4801).

In this example, the traffic flows through the **genev_sys_4800** interface.

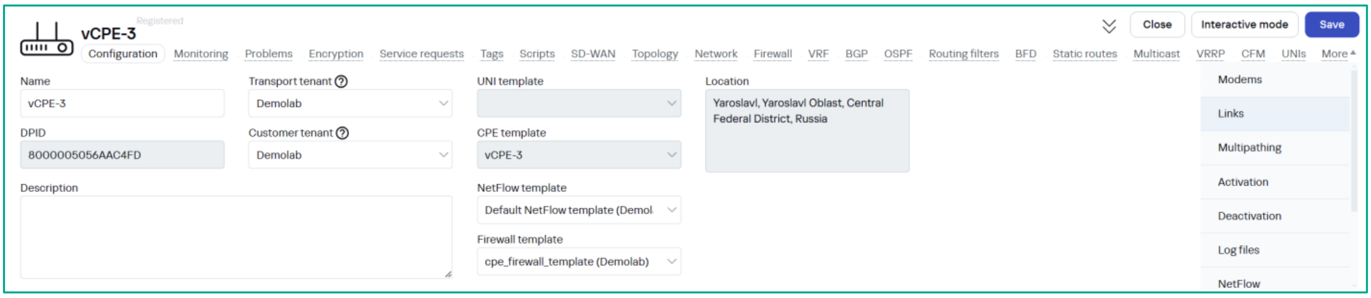


3.7.3. Set the Last resort parameter for vCPE-3 links.

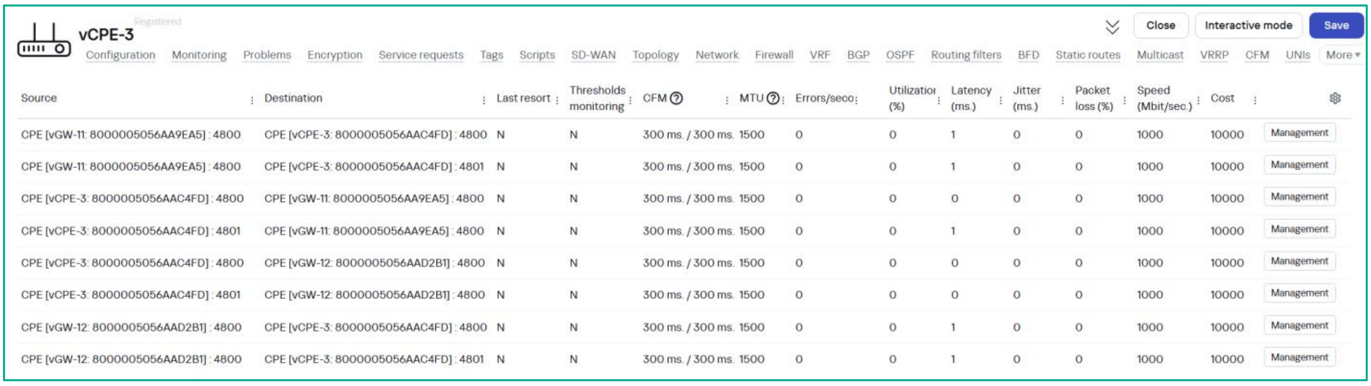
Go to the **CPE** menu and select **vCPE-3**.



Switch to the **Links** tab.



A list of links established with **vCPE-3** is displayed.



Source	Destination	Last resort	Thresholds monitoring	CFM	MTU	Errors/seco	Utilization (%)	Latency (ms.)	Jitter (ms.)	Packet loss (%)	Speed (Mbit/sec.)	Cost	
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vGW-11: 8000005056AA9EA5] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-11: 8000005056AA9EA5] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4800	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vCPE-3: 8000005056AAC4FD] : 4801	CPE [vGW-12: 8000005056AAD2B1] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	0	0	0	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4800	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management
CPE [vGW-12: 8000005056AAD2B1] : 4800	CPE [vCPE-3: 8000005056AAC4FD] : 4801	N	N	300 ms. / 300 ms.	1500	0	0	1	0	0	1000	10000	Management

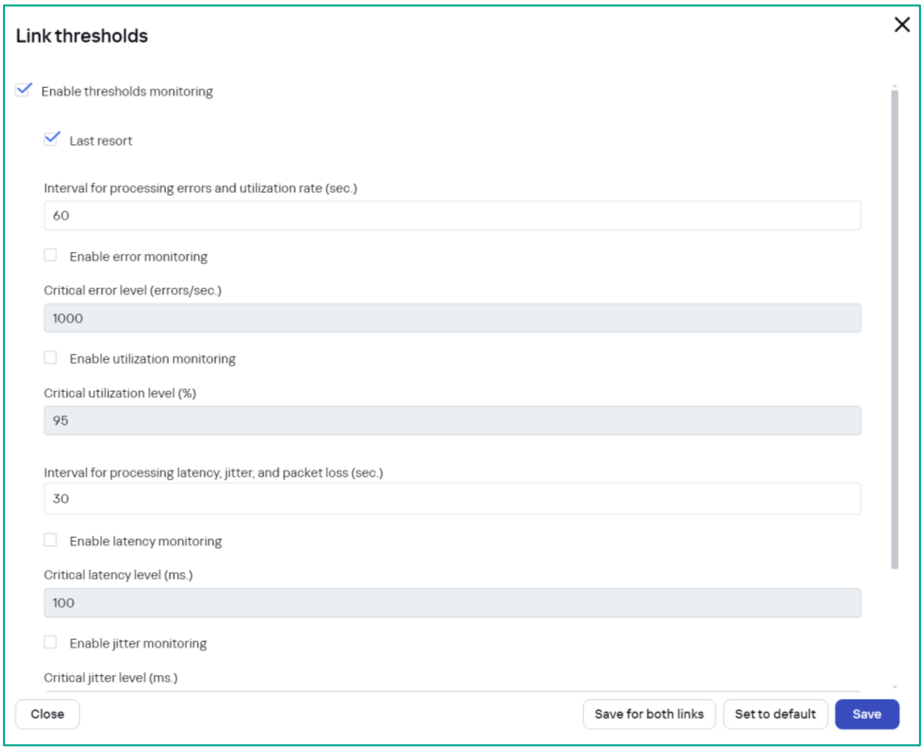
Find all links through which traffic flows: the source and destination ports of the links (4800 or 4801) must match the interface number according to the check in 3.7.2. In this example, the result of the check is that traffic flows through the **genev_sys_4800** link.

The links through which traffic flows in this example:

- **vCPE-3**:**4800** - vGW-11:4800
- **vCPE-3**:**4800** - vGW-12:4800
- vGW-11:4800 - **vCPE-3**:**4800**
- vGW-12:4800 - **vCPE-3**:**4800**

For each link with port 4800 for vCPE-3, click **Management → Set thresholds** to set **Last resort** parameter:

- Check **Enable tunnel thresholds monitoring**.
- Check **Last resort**.

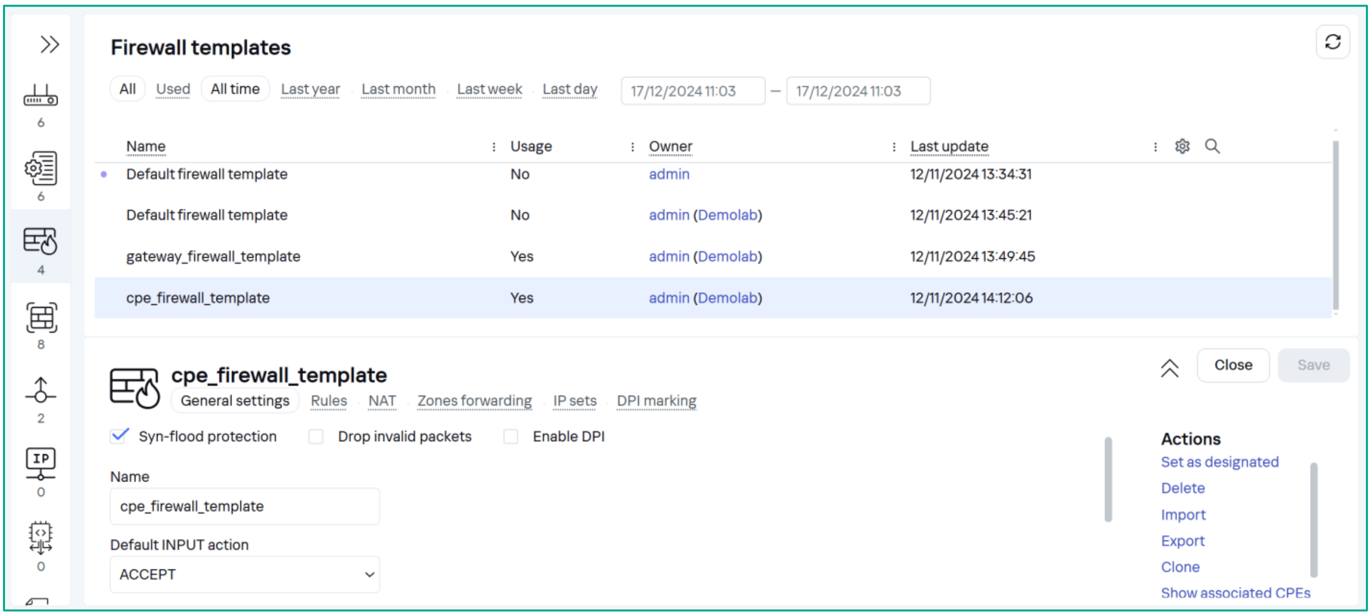


Click **Save for both links**.

3.7.4. Enable DPI in CPE firewall template.

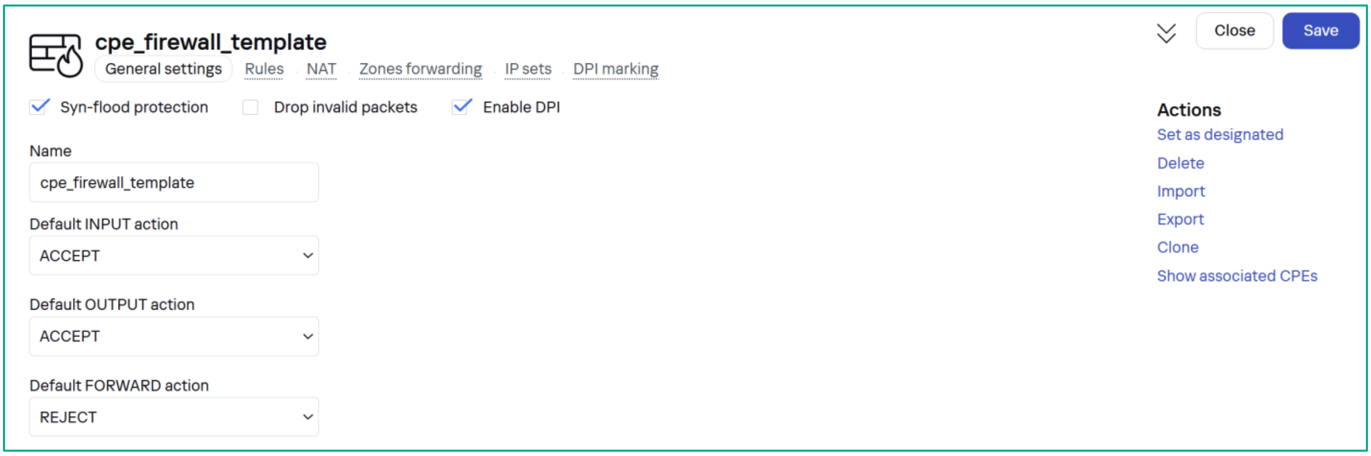
DPI must to be enabled in the firewall template settings.

Go to **Firewall templates** and select the template that applies to vCPE-3 and vCPE-4 (**cpe_firewall_template**).



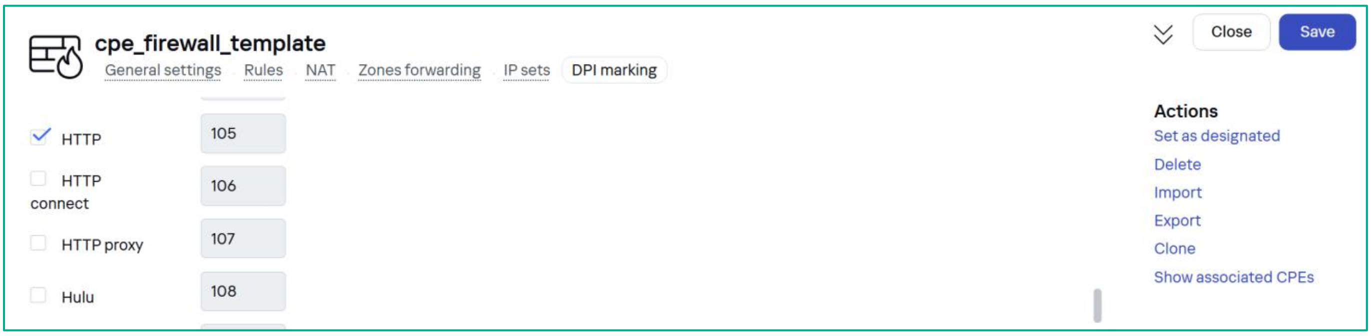
kaspersky

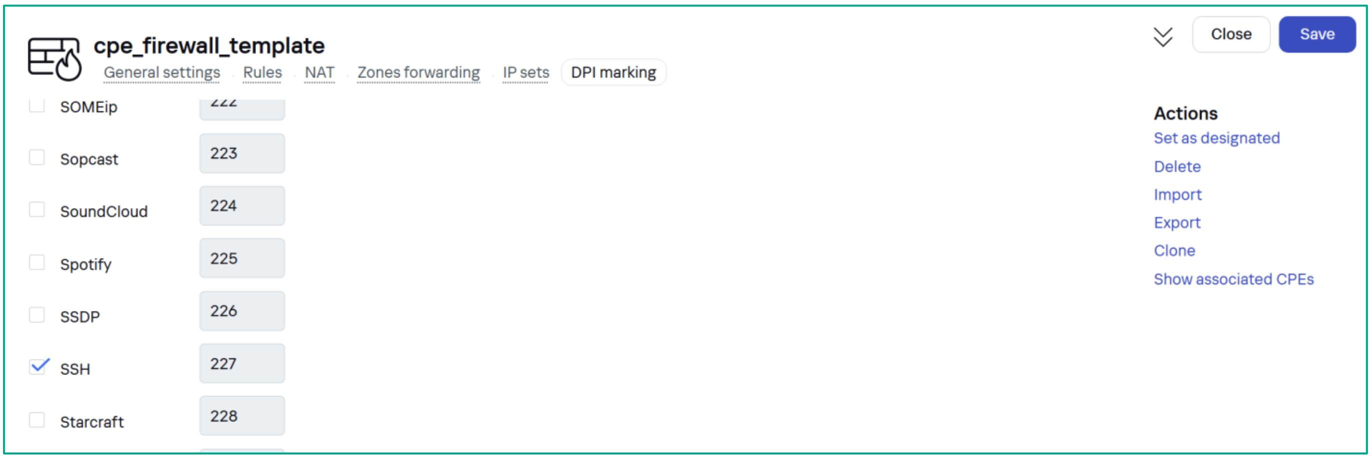
On the **General settings** tab check **Enable DPI**.



In the **DPI marking** tab, mark the protocols to be detected by DPI:

- **HTTP**
- **SSH**



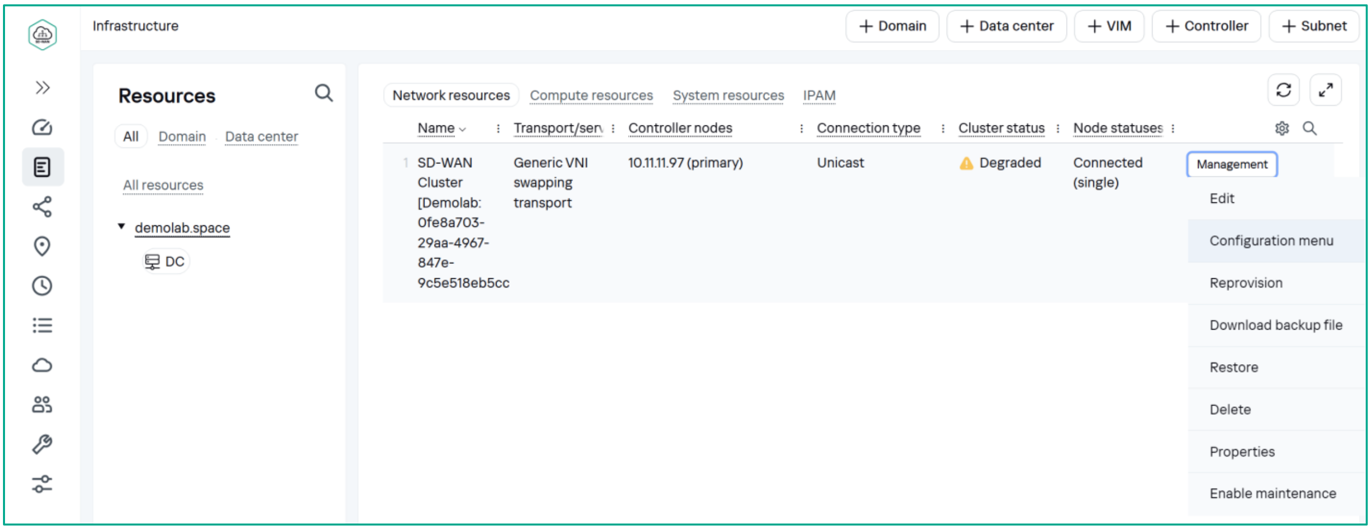


Click **Save**.

3.7.5. Create a constraint to exclude links with Last resort parameter set.

Constraints must be created to redirect traffic.

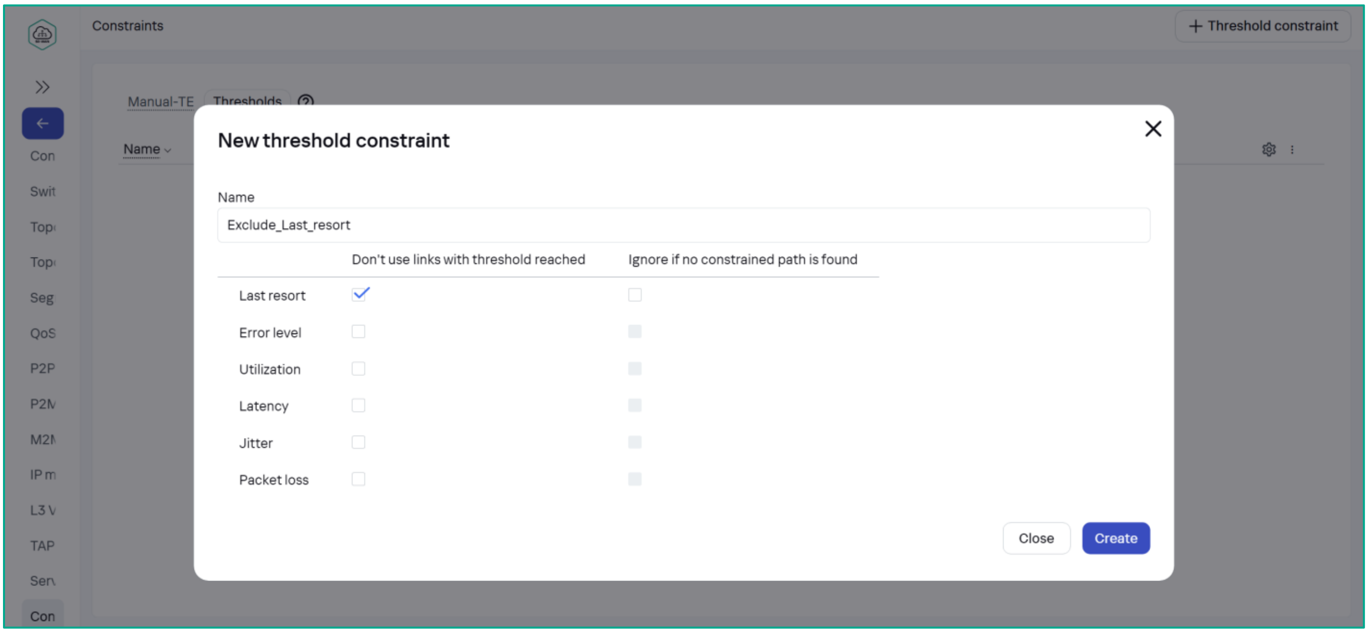
Go to **Infrastructure → SD-WAN cluster → Configuration menu**.



Go to the **Constraints** menu, then open the **Thresholds** tab and click the **+ Threshold Constraint** button.

Set constraint parameters:

- **Name**: **Exclude_Last_resort**.
- Check **Last resort**.



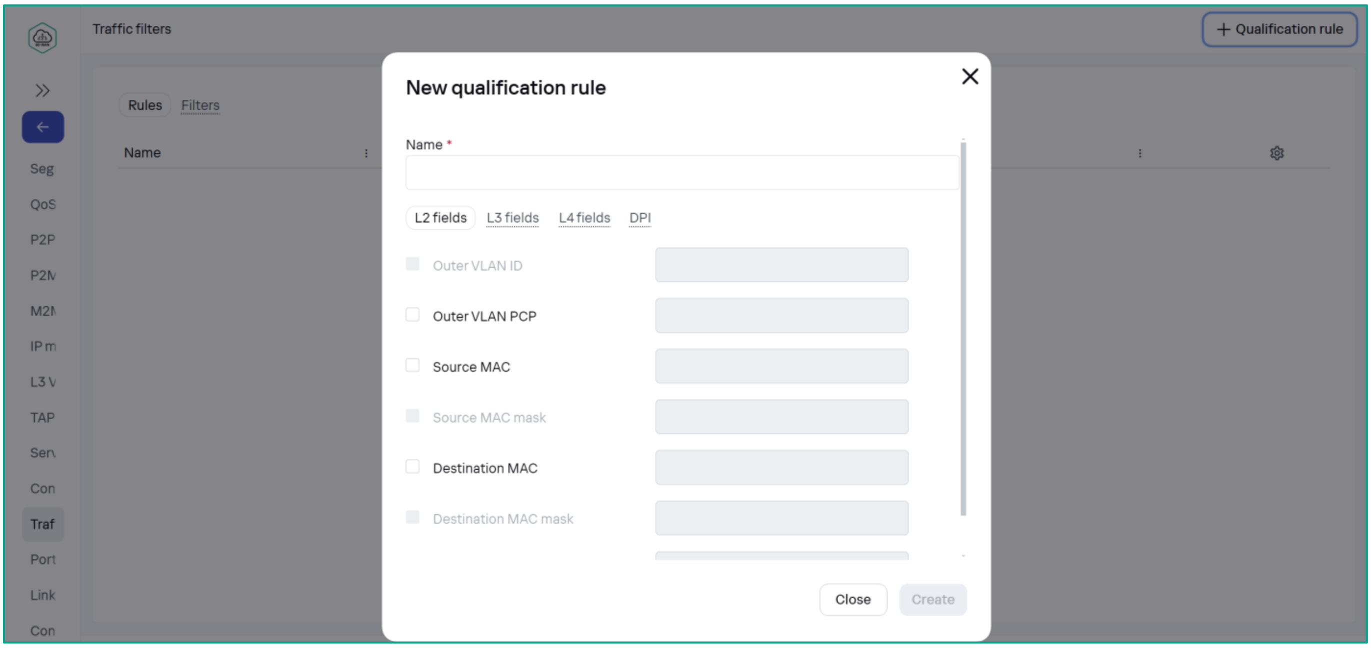
Click **Create**.

This constraint will exclude links marked as Last resort from the transport service.

3.7.6. Create traffic qualification rule for the SSH traffic.

To redirect traffic to a separate service, you must create a traffic filter with DPI classifier rules.

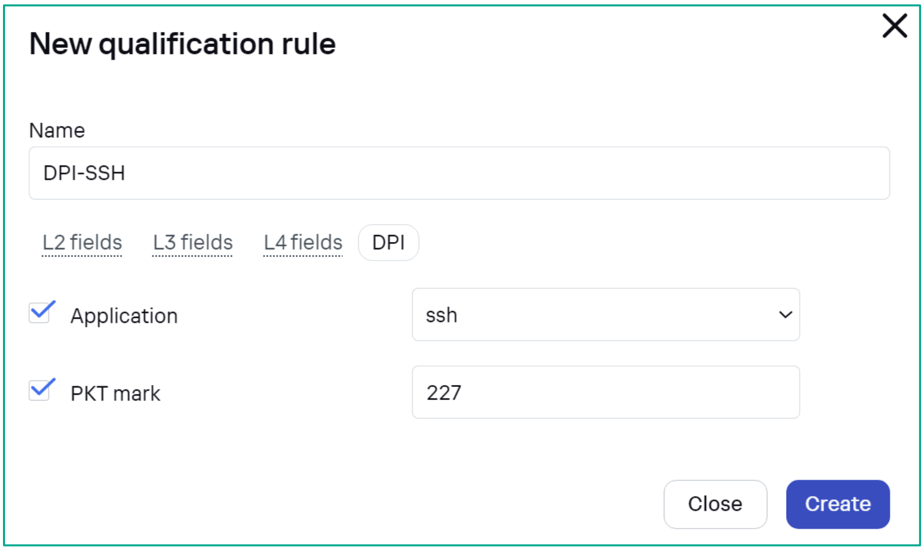
Go to the **Traffic Filters** tab in the menu. Then go to the **Rules** tab and click **+ Qualification rule**.



Set rule parameters:

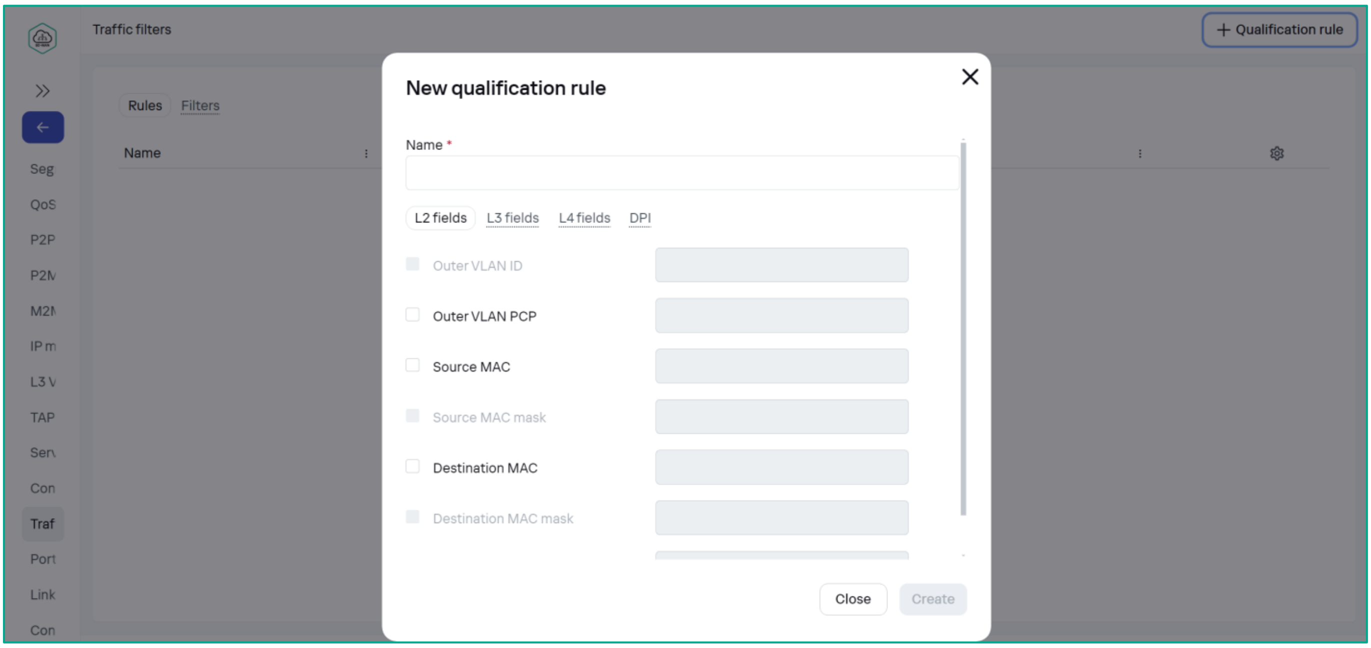
- **Name**: **DPI-SSH**
- **L3 Fields**:
    - **Protocol**: **IPv4**
- **DPI**:
    - **Application**: **ssh**

Click **Create**.



3.7.7. Create traffic qualification rule for the HTTP traffic.

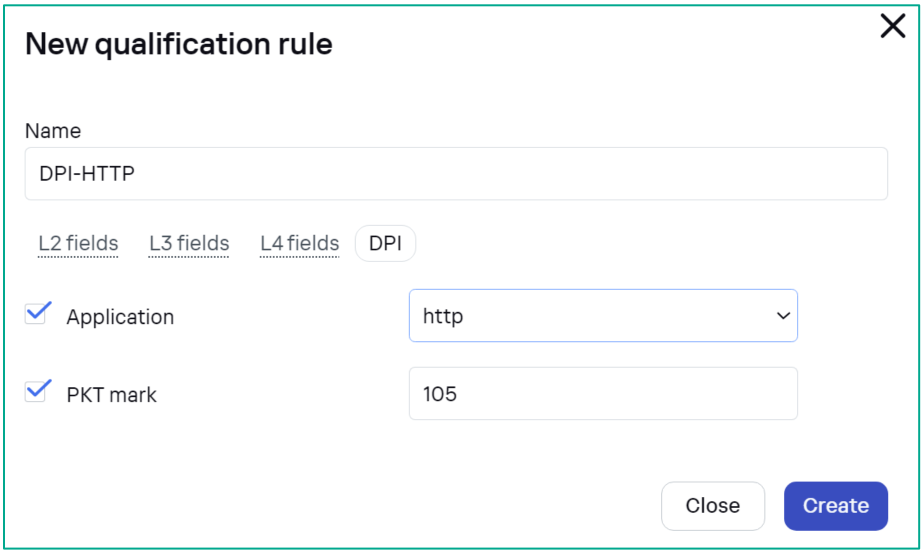
Go to the **Traffic Filters** tab in the menu. Then go to the **Rules** tab and click **+ Qualification rule**.



Set rule parameters:

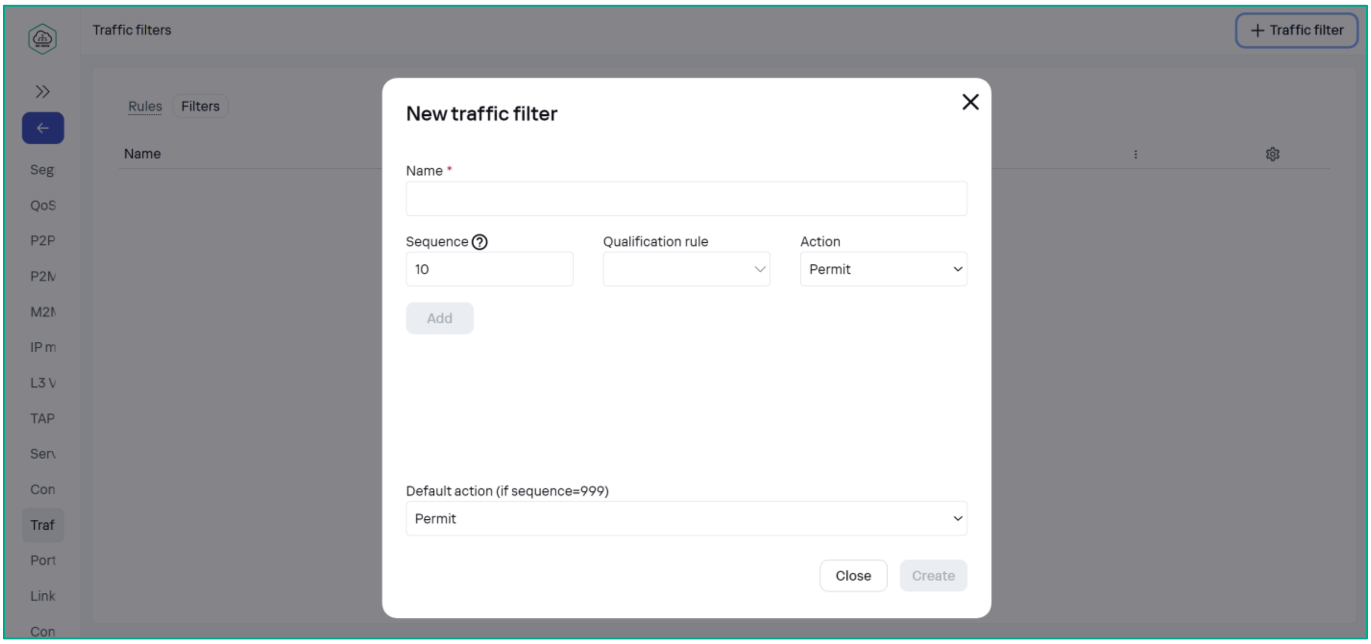
- **Name**: **DPI-HTTP**
- **L3 Fields**:
  - **Protocol**: **IPv4**
- **DPI**:
  - **Application**: **http**

Click **Create**.



3.7.8. Create a traffic filter to redirect test traffic to a separate service.

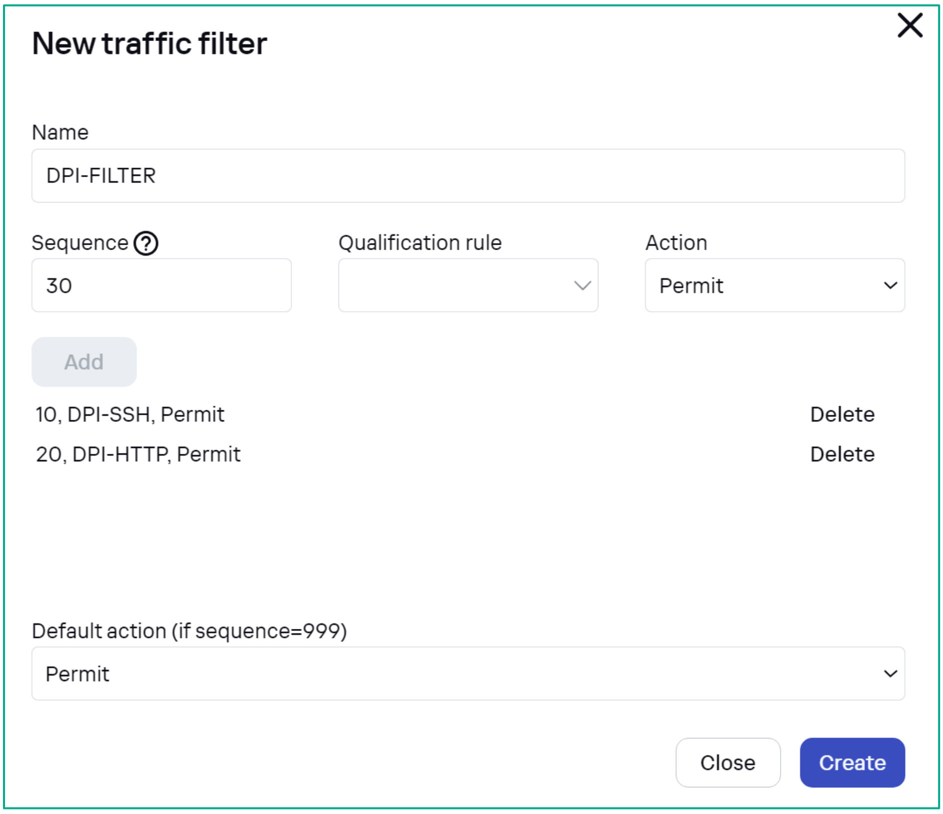
Go to **Traffic filters** tab, click + **Traffic filter**.



Set traffic filter parameters:

- **Name**: **DPI-FILTER**.
- Add **Qualification rules**: Select **rules**, created in 3.7.6 and 3.7.7 (**DPI-SSH** and **DPI-HTTP**), set **Action**: **Permit**. Click **Add**.

Click **Create**.



3.7.9. Create ACL Service interfaces.

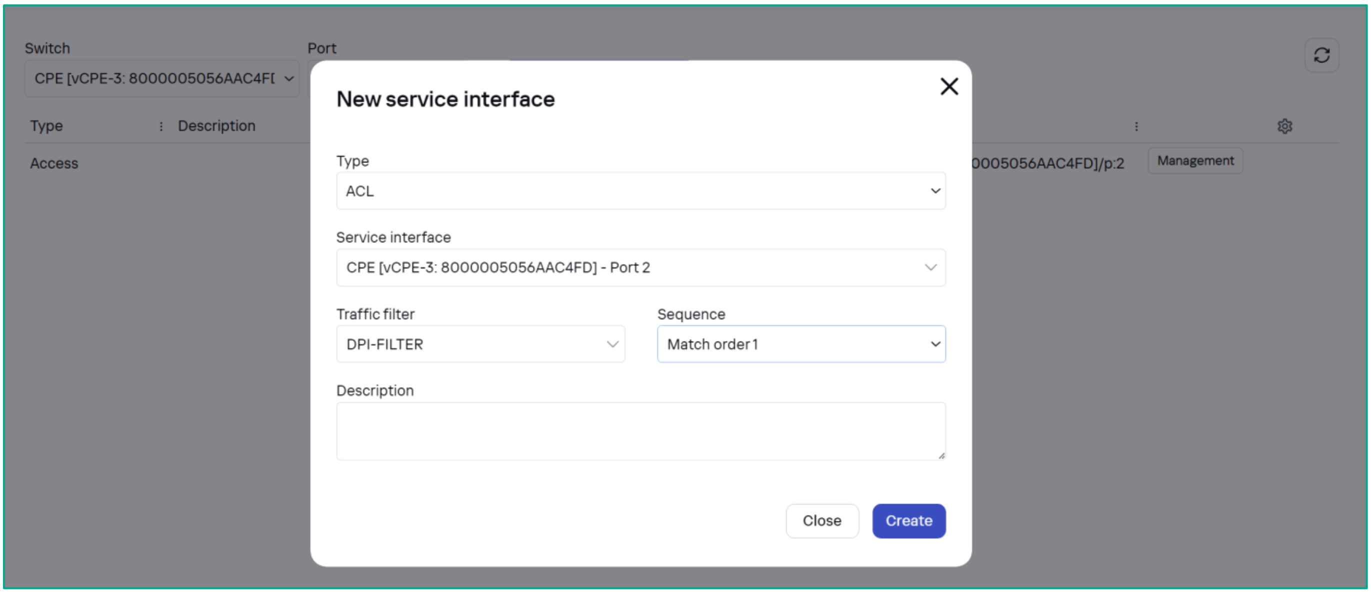
Traffic enters the transport service through service interfaces. It is necessary to create a special ACL Service Interface (ACL SI). Go to the **Service Interfaces** tab and select **Switch**: **vCPE-3** and **Port**: **2 (ovs-lan).**

Click **Create service interface**.

Set service interface parameters:

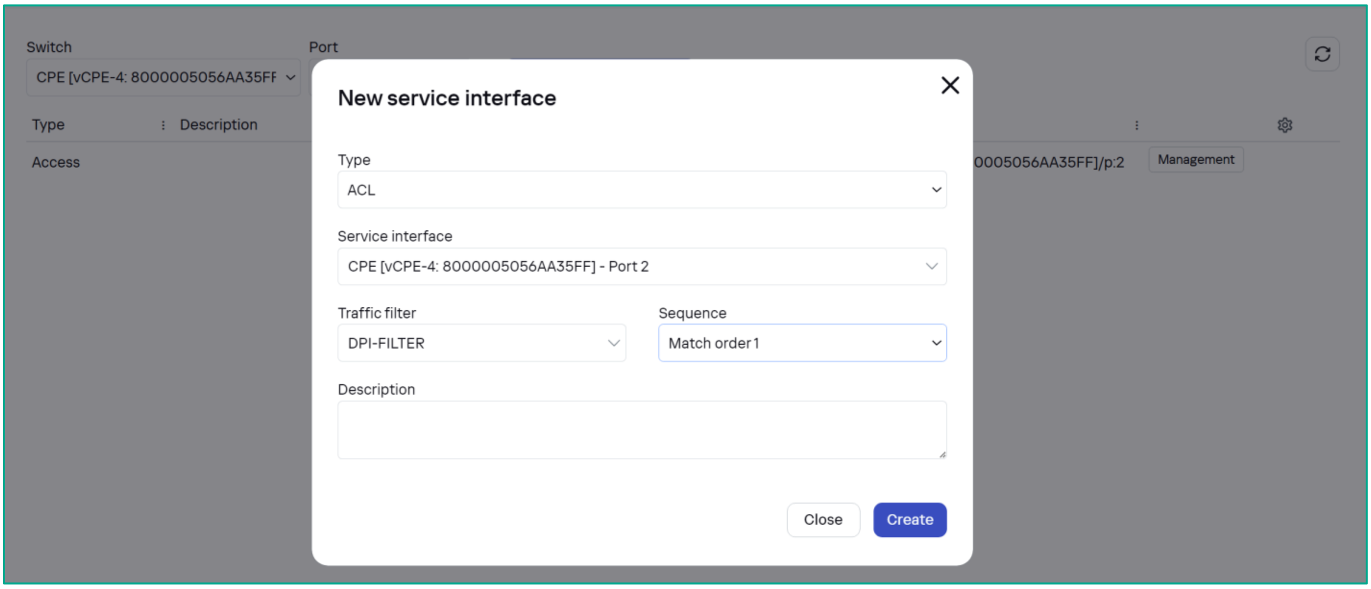
- **Type**: **ACL**.
- **Service interface**: **vCPE-3** - **Port 2**.
- **Traffic Filter**: traffic filter created in step 3.7.8 (**DPI-FILTER**).
- **Sequence**: **Match order 1** (this ACL SI will be the first to process traffic).

Click **Create**.



To create a new transport service, you must create service interfaces for each CPE.

Create a similar ACL service interface for **vCPE-4**.



3.7.10. Create a separate transport service for priority traffic.

Go to the **M2M Services** menu, click **+ M2M service**.

Set service parameters:

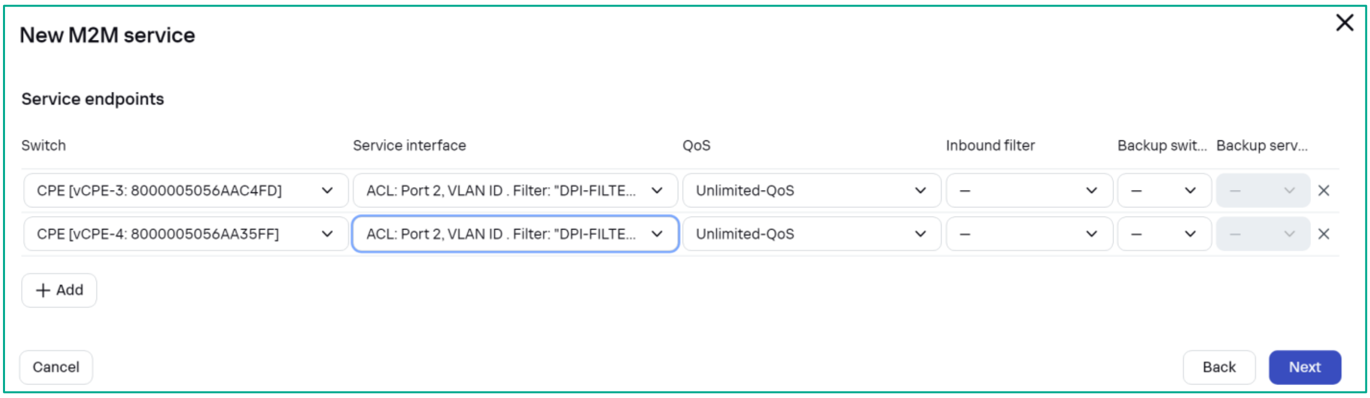
- **Name**: **M2M_ACL**.
- **Constraint**: **Threshold** created in 3.7.5 (**Exclude_Last_resort**).

Click **Next**.

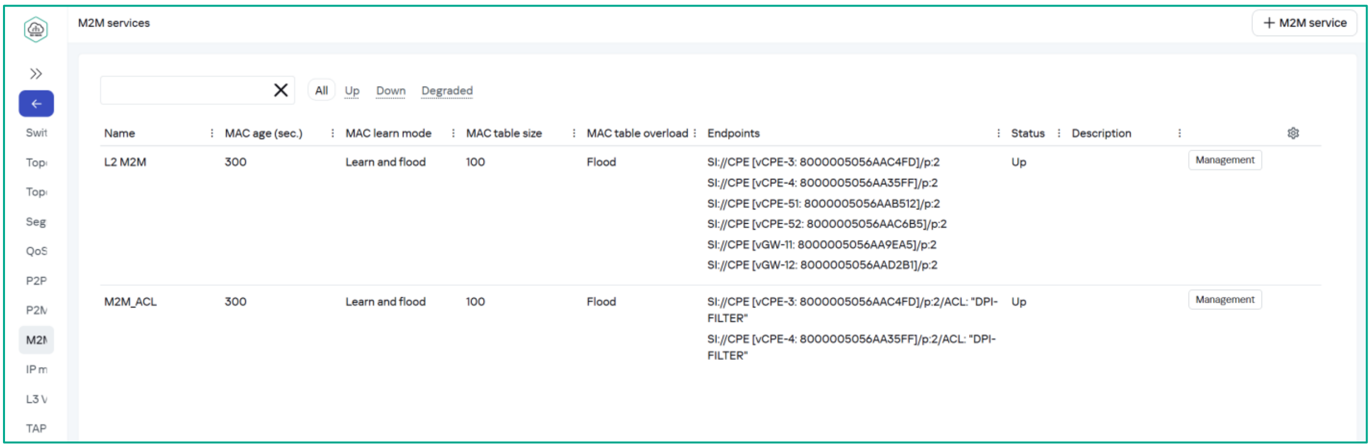
Click **+ Add** in **Service endpoints** section to add service interfaces created in 3.7.9.

Set **service endpoints** parameters:

- **Switch**: **vCPE-3** and **vCPE-4**.
- **Service interface**: created in 3.7.9 **ACL Service Interfaces** for vCPE-3 and vCPE-4.
- **QoS**: **Unlimited QoS**.



Click **Next** and **Create**.



3.7.11. Verify SSH traffic redirection.

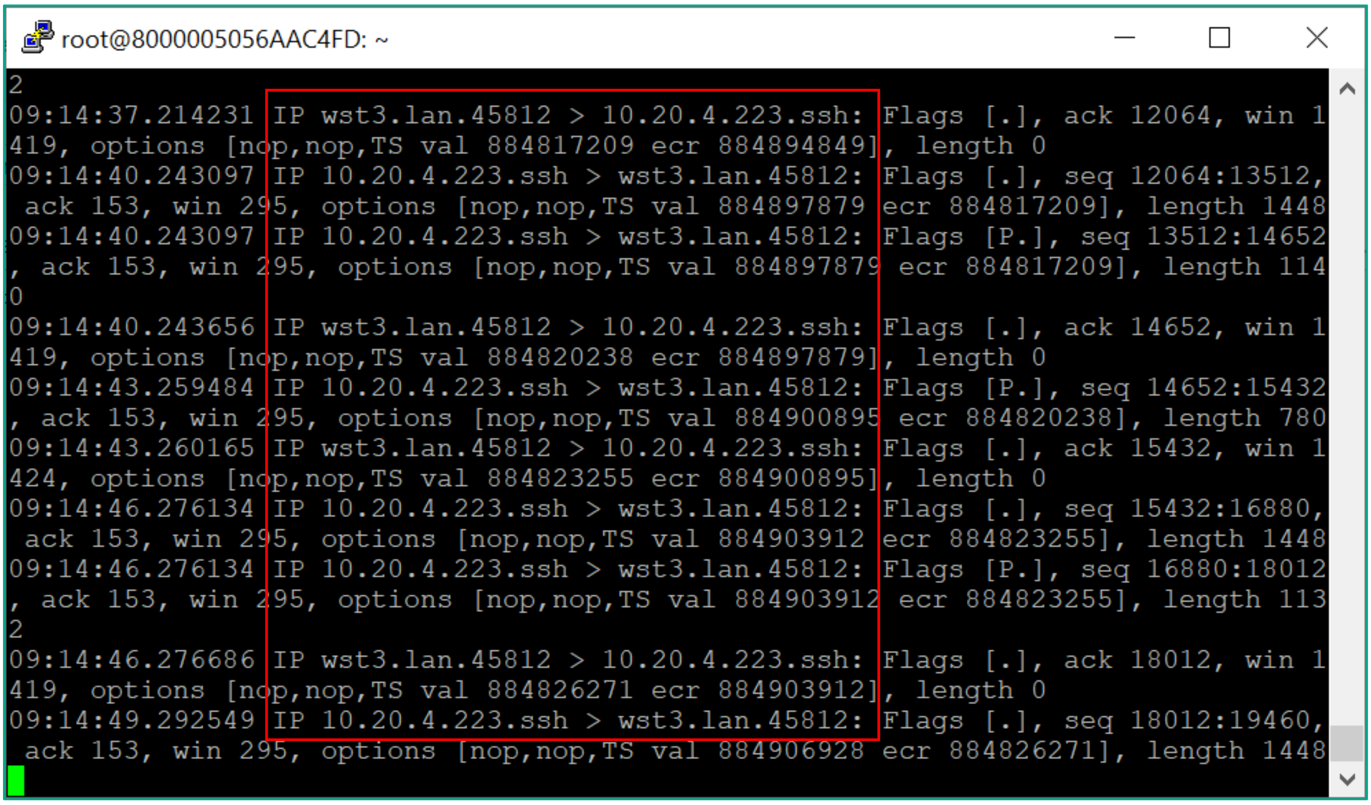
Connect to vCPE-3 and verify that the traffic is redirected to a second WAN interface (depending on the settings made earlier):

In 3.7.2 it was verified that the traffic flows through the interface **genev_sys_4800** (sdwan0). After configuring a separate transport service due to constraints and filtering, the traffic was redirected to interface **genev_sys_4801** (sdwan1).

Use tcpdump to check if there is traffic on the **genev_sys_4801** interface:

```
tcpdump -i genev_sys_4801
```

The screenshot shows that traffic has switched from interface **genev_sys_4800** (sdwan0) to **genev_sys_4801** (sdwan1).



```
root@8000005056AAC4FD: ~
2
09:14:37.214231 IP wst3.lan.45812 > 10.20.4.223.ssh: Flags [.], ack 12064, win 1
419, options [nop,nop,TS val 884817209 ecr 884894849], length 0
09:14:40.243097 IP 10.20.4.223.ssh > wst3.lan.45812: Flags [.], seq 12064:13512,
 ack 153, win 295, options [nop,nop,TS val 884897879 ecr 884817209], length 1448
09:14:40.243097 IP 10.20.4.223.ssh > wst3.lan.45812: Flags [P.], seq 13512:14652
, ack 153, win 295, options [nop,nop,TS val 884897879 ecr 884817209], length 114
0
09:14:40.243656 IP wst3.lan.45812 > 10.20.4.223.ssh: Flags [.], ack 14652, win 1
419, options [nop,nop,TS val 884820238 ecr 884897879], length 0
09:14:43.259484 IP 10.20.4.223.ssh > wst3.lan.45812: Flags [P.], seq 14652:15432
, ack 153, win 295, options [nop,nop,TS val 884900895 ecr 884820238], length 780
09:14:43.260165 IP wst3.lan.45812 > 10.20.4.223.ssh: Flags [.], ack 15432, win 1
424, options [nop,nop,TS val 884823255 ecr 884900895], length 0
09:14:46.276134 IP 10.20.4.223.ssh > wst3.lan.45812: Flags [.], seq 15432:16880,
 ack 153, win 295, options [nop,nop,TS val 884903912 ecr 884823255], length 1448
09:14:46.276134 IP 10.20.4.223.ssh > wst3.lan.45812: Flags [P.], seq 16880:18012
, ack 153, win 295, options [nop,nop,TS val 884903912 ecr 884823255], length 113
2
09:14:46.276686 IP wst3.lan.45812 > 10.20.4.223.ssh: Flags [.], ack 18012, win 1
419, options [nop,nop,TS val 884826271 ecr 884903912], length 0
09:14:49.292549 IP 10.20.4.223.ssh > wst3.lan.45812: Flags [.], seq 18012:19460,
 ack 153, win 295, options [nop,nop,TS val 884906928 ecr 884826271], length 1448
```

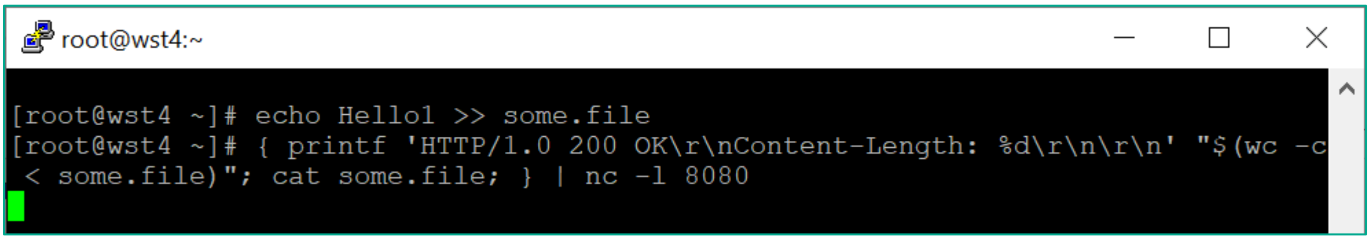
Note: In this scenario, all SSH and HTTP traffic from wst3 and wst4 is redirected to a separate service where vCPE3 and vCPE4 are added. Therefore, only addresses from these CPEs will be available from wst3 and wst4 via SSH and HTTP.

3.7.12. Verify HTTP traffic redirection.

To start temporary HTTP server, you can use the **nc** on the **wst4** host:

```
echo Hello1 >> some.file

{ printf 'HTTP/1.0 200 OK\r\nContent-Length: %d\r\n\r\n' "$(wc -c < some.file)"; cat some.file; } | nc -l 8080
```



```
root@wst4:~
[root@wst4 ~]# echo Hello1 >> some.file
[root@wst4 ~]# { printf 'HTTP/1.0 200 OK\r\nContent-Length: %d\r\n\r\n' "$(wc -c
 < some.file)"; cat some.file; } | nc -l 8080
```

To generate an HTTP request, open an HTTP session from **wst3** to port **8080** on **wst4**. For example, using **curl**:

```
curl <wst4 IP address>:8080
```

```
root@wst3:~
[root@wst3 ~]# curl 10.20.4.223:8080
Hello1
[root@wst3 ~]#
```

In the example below, you can see that HTTP traffic has been switched from the interface **genev_sys_4800** (sdwan0) to **genev_sys_4801** and DPI has detected HTTP traffic on the non-standard port.

```
root@8000005056AAC4FD: ~
09:25:56.220005 IP wst3.lan.41942 > 10.20.4.223.8080: Flags [P.], seq 2121946566
:2121946646, ack 2027220581, win 229, options [nop,nop,TS val 885496215 ecr 8855
73855], length 80: HTTP: GET / HTTP/1.1
09:25:56.221627 IP 10.20.4.223.8080 > wst3.lan.41942: Flags [P.], seq 1:54, ack
0, win 227, options [nop,nop,TS val 885573858 ecr 885496214], length 53: HTTP: H
TTP/1.0 200 OK
09:25:56.221696 IP 10.20.4.223.8080 > wst3.lan.41942: Flags [.], ack 80, win 227
, options [nop,nop,TS val 885573858 ecr 885496215], length 0
09:25:56.221696 IP 10.20.4.223.8080 > wst3.lan.41942: Flags [F.], seq 54, ack 0,
 win 227, options [nop,nop,TS val 885573858 ecr 885496214], length 0
09:25:56.222142 IP wst3.lan.41942 > 10.20.4.223.8080: Flags [.], ack 54, win 229
, options [nop,nop,TS val 885496217 ecr 885573858], length 0
09:25:56.222231 IP wst3.lan.41942 > 10.20.4.223.8080: Flags [F.], seq 80, ack 55
, win 229, options [nop,nop,TS val 885496217 ecr 885573858], length 0
09:25:56.223344 IP 10.20.4.223.8080 > wst3.lan.41942: Flags [.], ack 81, win 227
, options [nop,nop,TS val 885573860 ecr 885496217], length 0
```

3.7.13. Restore the settings after the test is completed.

Delete the **transport service** created in 3.7.10 (when deleting, check **Delete associated service interfaces**).

Remove the **Last resort** parameter added in 3.7.3 from **vCPE-3** links.

Stop **SSH** session from **wst3** to **wst4**, started in 3.7.1.

Remove **DPI** configuration from **CPE firewall template**, added in 3.7.4.

4. SD-WAN Topology Configuration

Links form a topology that determines the connectivity of devices in the data plane and is responsible for optimizing the passage of traffic of transport services. In Kaspersky SD-WAN, devices can be arranged in one of the following topologies:

- **Hub-and-Spoke** is the default topology in which links between CPE devices are established through the SD-WAN Gateway.
- **Full-Mesh** is a topology in which direct links are created between all CPE devices.
- **Partial-Mesh** is a topology in which direct links are established between some of the CPE devices.

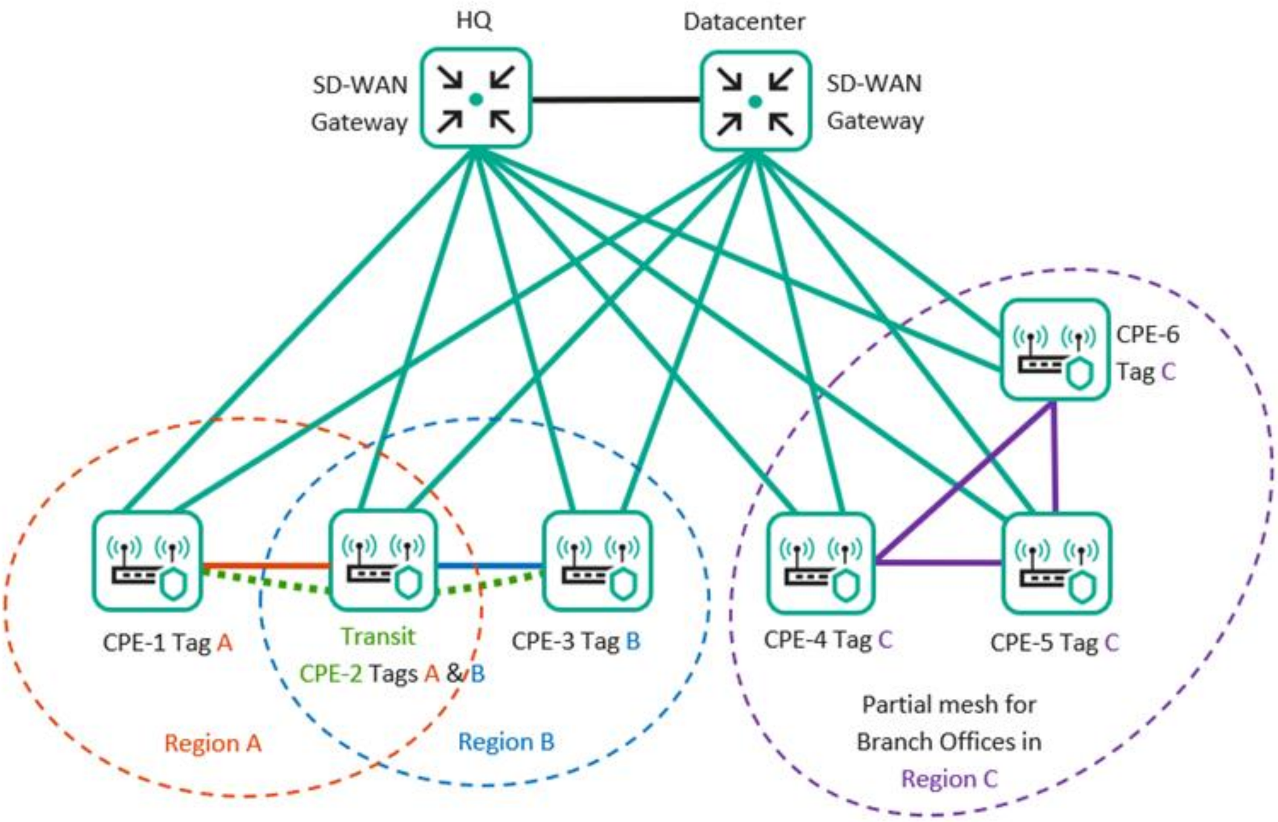


Figure 4.1 SD-WAN network topologies

To build network topologies, the Kaspersky SD-WAN solution uses topology tags assigned to CPE devices.

A CPE device can also be a transit device. In this case, other CPE devices can establish segments through it.

For more information, please refer to Kaspersky SD-WAN Online Help:
https://support.kaspersky.com/help/SD-WAN/2.3/en-US/250984.htm

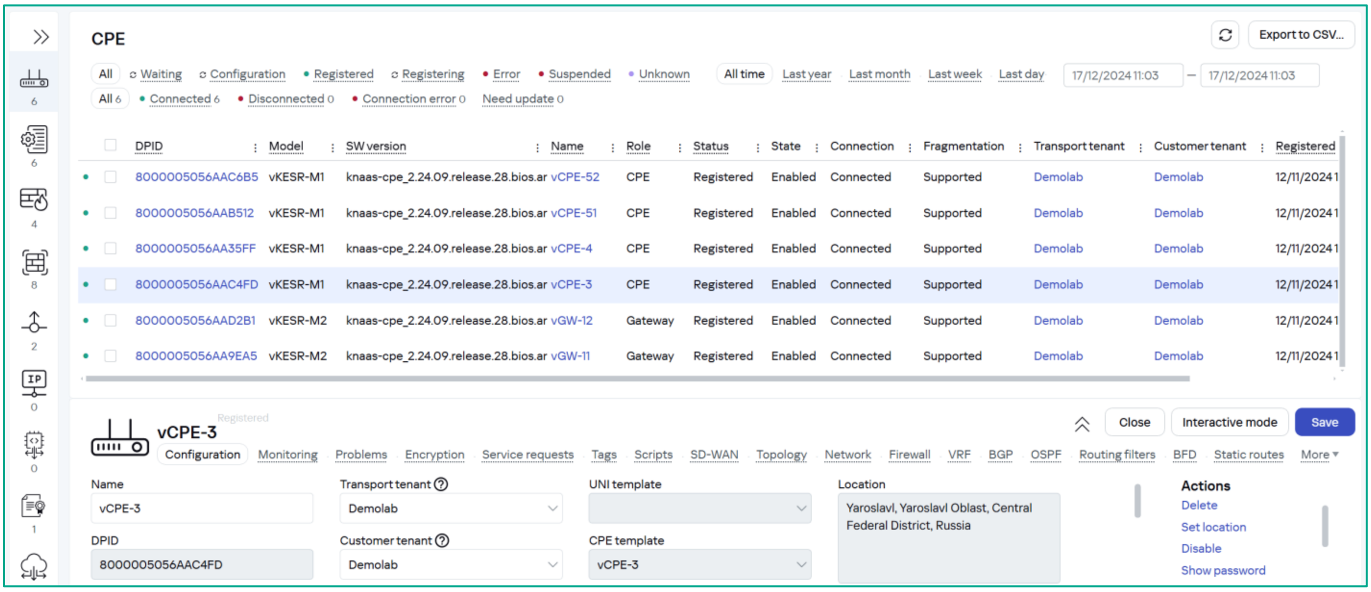
4.1. **Creating Full-Mesh topology**

This scenario configures Full-Mesh topologies between CPE devices by adding the same topology tag to CPE devices. The built topology will be displayed in the transport service settings. The additionally constructed paths between CPE devices will also be displayed in the Segments section.

4.1.1. Assign Topology Tags to the CPE devices.

To create a Full-Mesh topology, the CPEs must have the same topology tags.

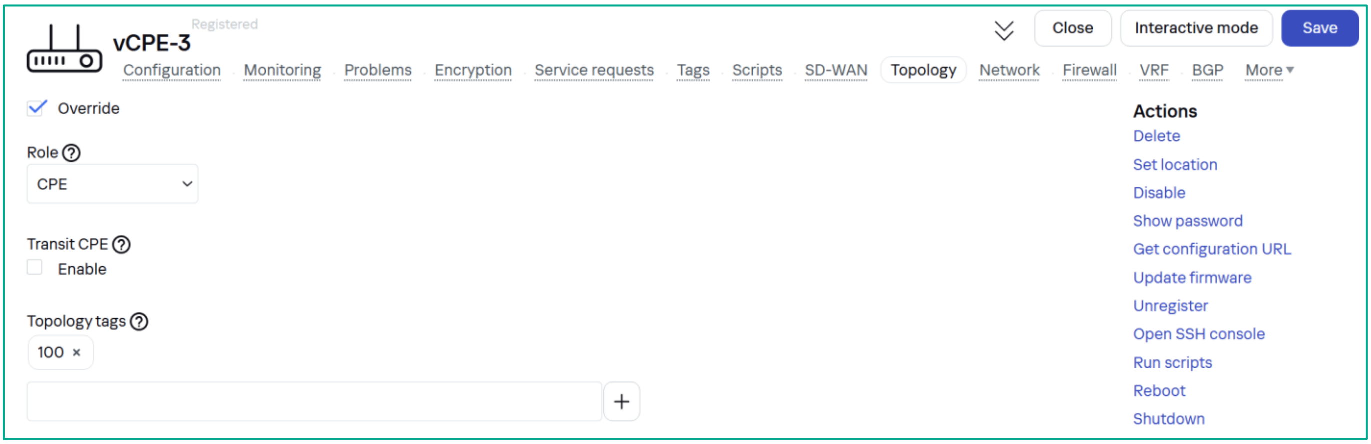
To configure them, go to the **CPE** menu and select **vCPE-3**.



Go to the **Topology** tab.

Set topologies parameters:

- Check **Override**.
- Add tag **100** (click **+** to add).

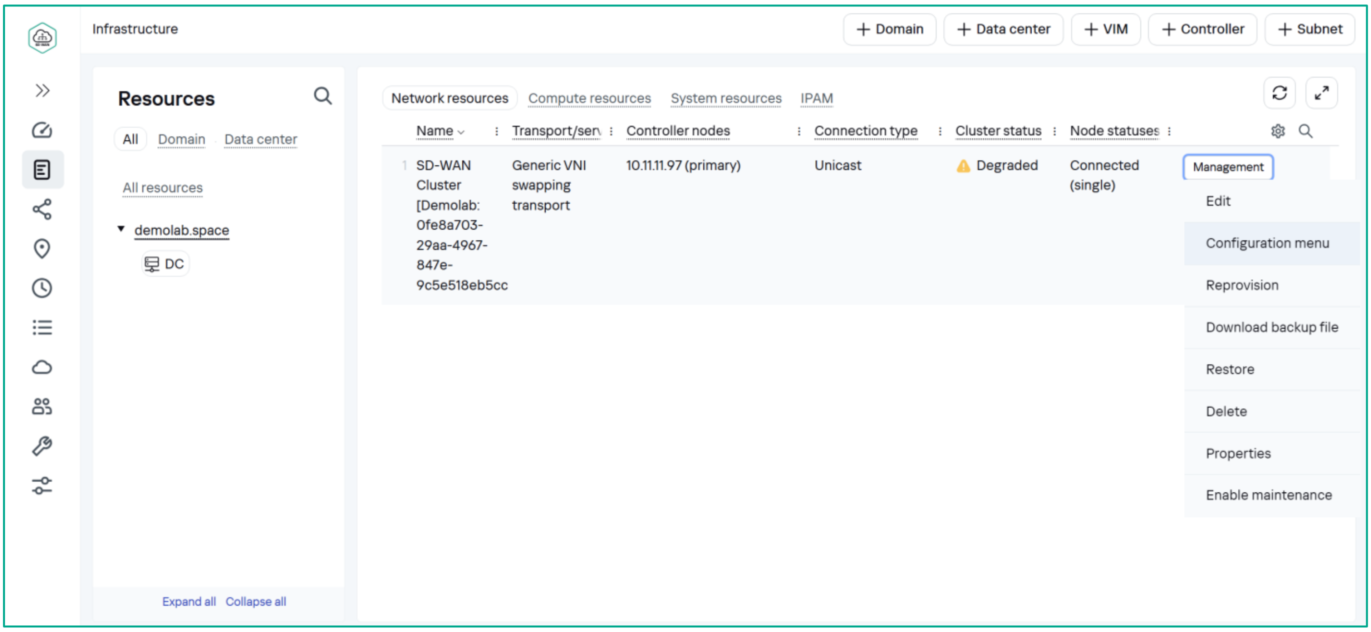


Click **Save** (the orchestrator will apply new settings to the CPE).

Assign topology tag **100** to the **vCPE-4**, **vCPE-51** and **vCPE-52** devices.

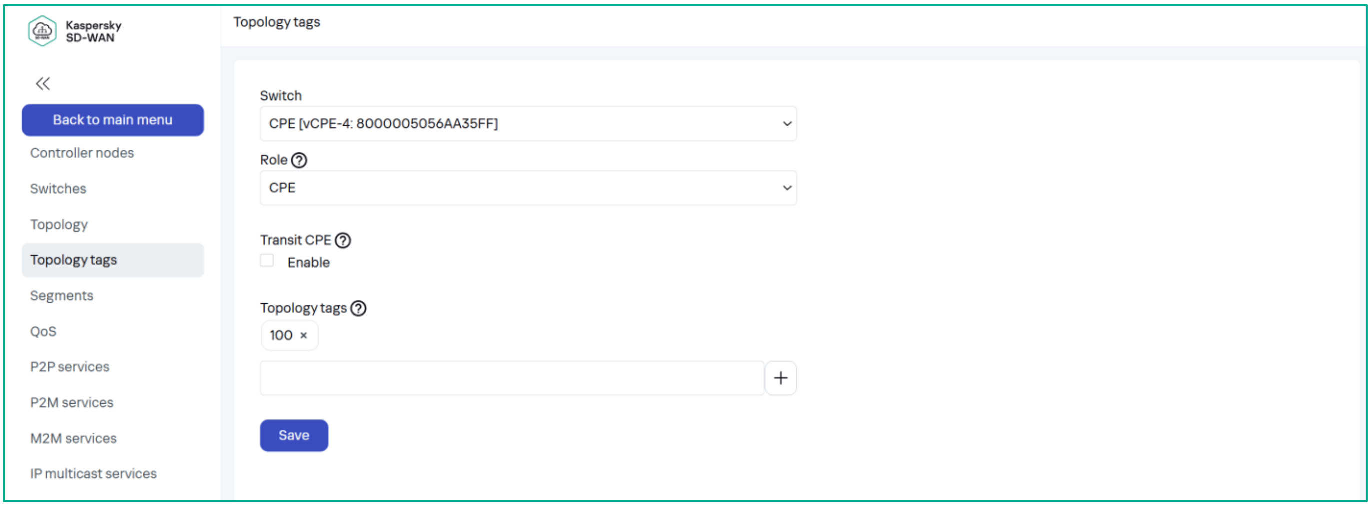
Alternatively, you can assign topology tags in the controller settings.

Go to **Infrastructure → SD-WAN cluster → Configuration menu**.



Open **Topology tags** menu**.**

Select **CPE** in the **Switch** selector and then add a topology tag (click **+**), then click **Save** to apply new settings.

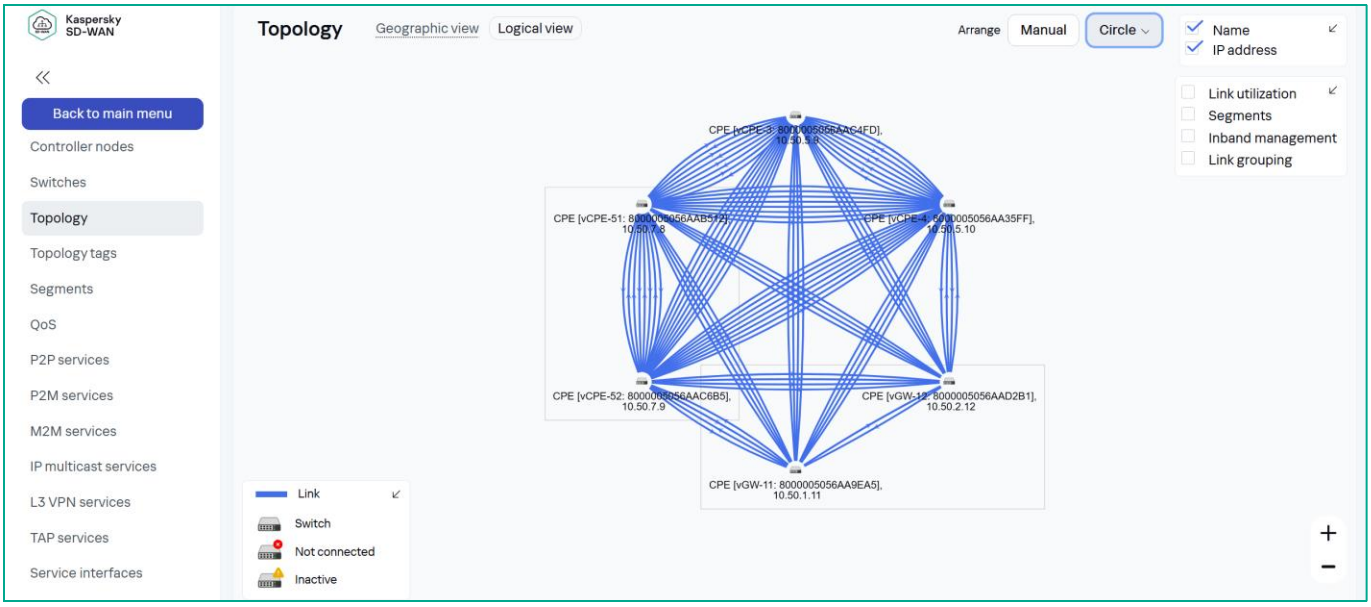


4.1.2. Display the built topology.

To view the built topology, go to **Infrastructure → SD-WAN cluster → Configuration menu → Topology**.

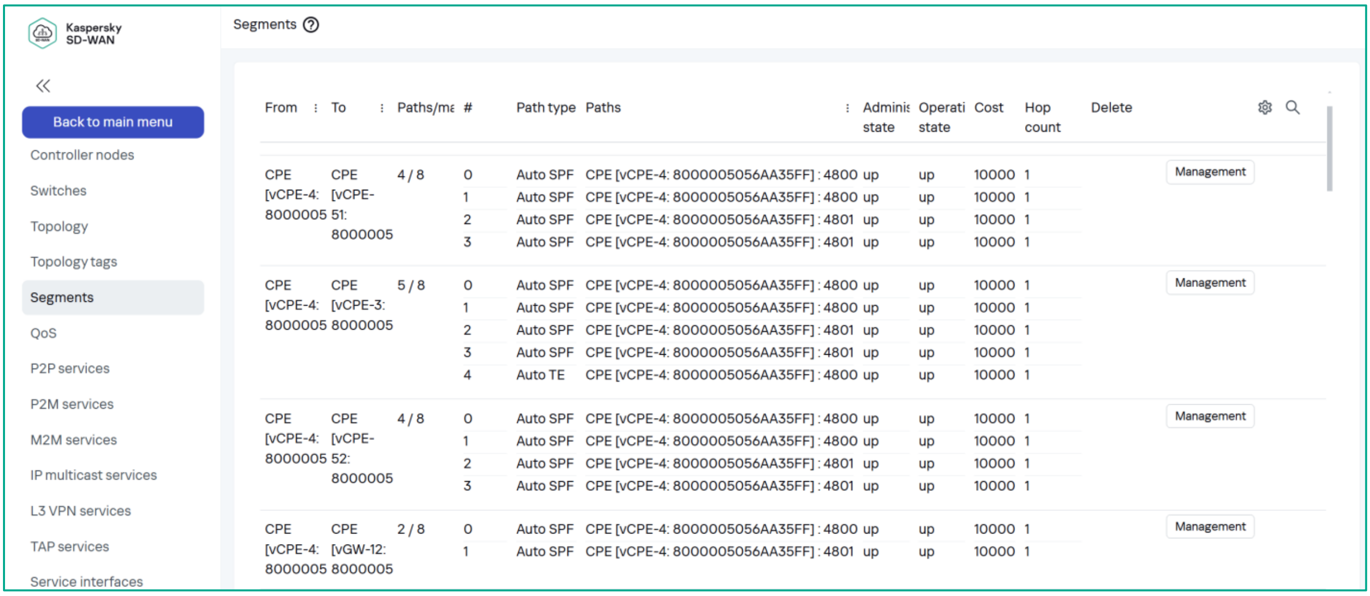
Open **Logical view** tab. Select **Arrange**: **Circle** for a more convenient view.

The built service topology will be displayed. The screenshot shows Full-Mesh topology between CPEs, also CPE devices have retained links to vGW-11/12 devices (gateways).



To check the segments between CPE devices, go to the **Segments** tab.

In the screenshot below, you can see that the new links between vCPE devices that do not pass through gateways (vGW11/12) have been created. As you can see, the vCPEs have formed a Full-Mesh topology.



4.1.3. Restore the settings after the test is completed.

Remove topology tags from CPE devices added in 4.1.1.

kaspersky

4.2. **Creating Partial-Mesh topology**

This scenario configures Partial-Mesh topologies between CPE devices. Two groups of CPE devices will be created:

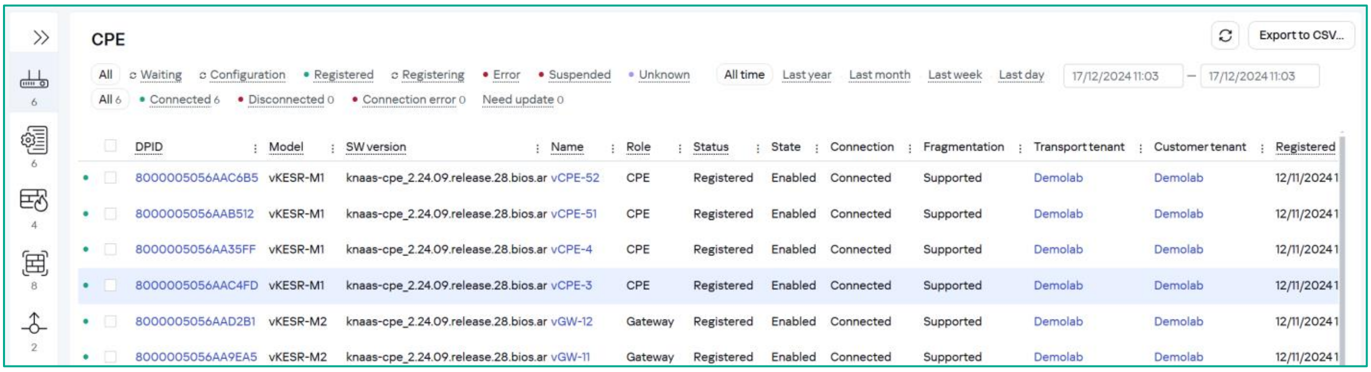
- vCPE-3 and vCPE-4.
- vCPE-51, vCPE-52 and vCPE-4.

To build the Partial-Mesh topology, topology tags will be assigned to CPE devices, separately for each group. The built topology will be displayed in the transport service settings. The additionally constructed paths between CPE devices will also be displayed in the Segments section.

4.2.1. Assign Topology Tags to the CPE devices.

To create a Partial-Mesh topology, you must assign different topology tags to the CPE devices according to the desired topology.

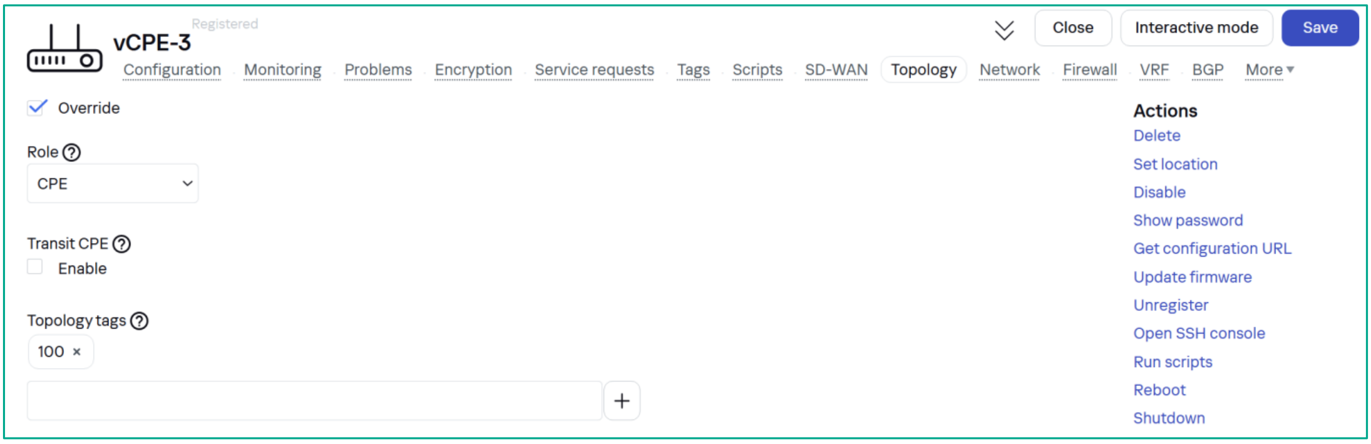
To configure them, go to the **CPE** menu and select **vCPE-3**.



Go to the **Topology** tab.

Set topologies parameters:

- Check **Override**.
- Add tag **100** (click **+** to add).



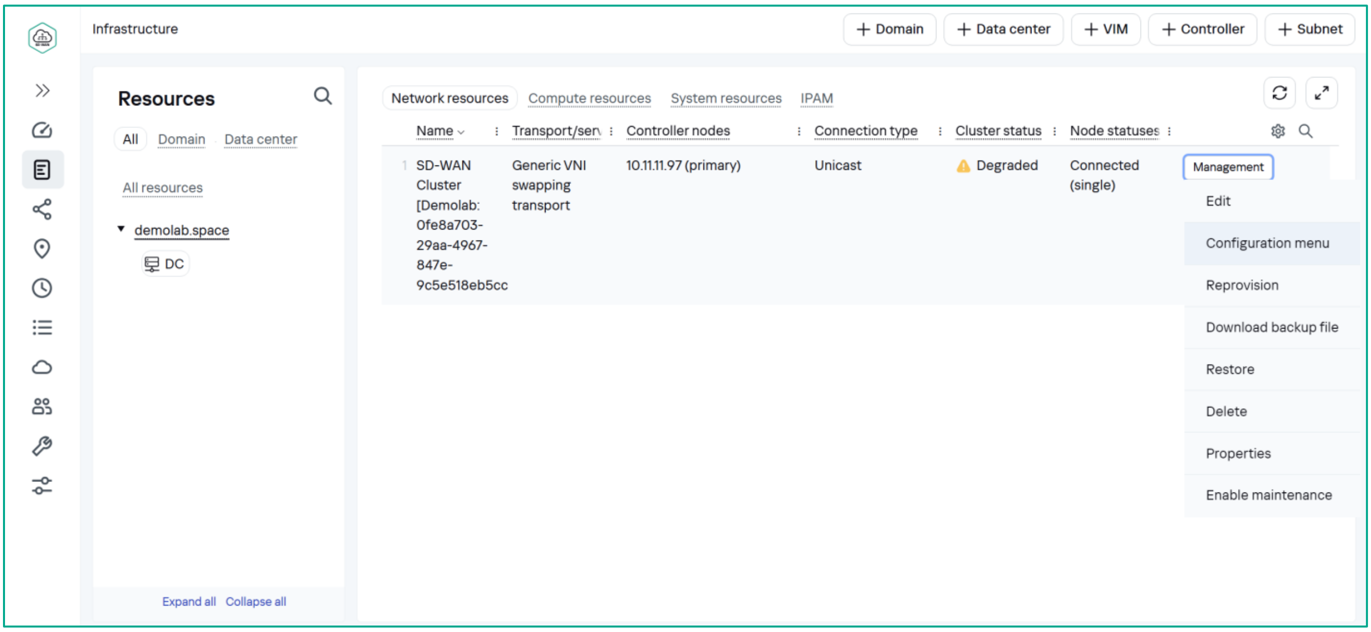
Click **Save** (the orchestrator will apply new settings to the CPE).

Assign topology tags to other CPE devices:

- **vCPE-51**: **200**.
- **vCPE-52**: **200**.
- **vCPE-4**: **100** and **200**.

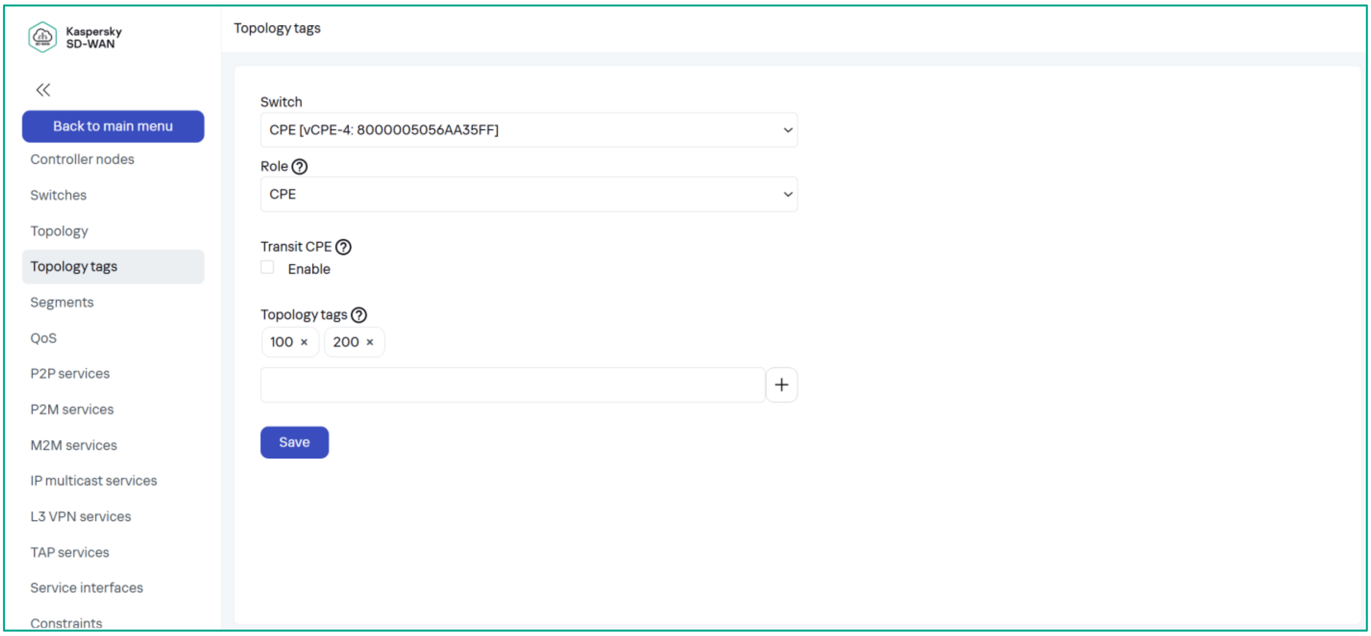
Alternatively, you can assign topology tags in the controller settings.

Go to **Infrastructure → SD-WAN cluster → Configuration menu**.



Open **Topology tags** menu**.**

Select **CPE** in the **Switch** selector and then add a topology tags (click **+**), then click **Save** to apply new settings.



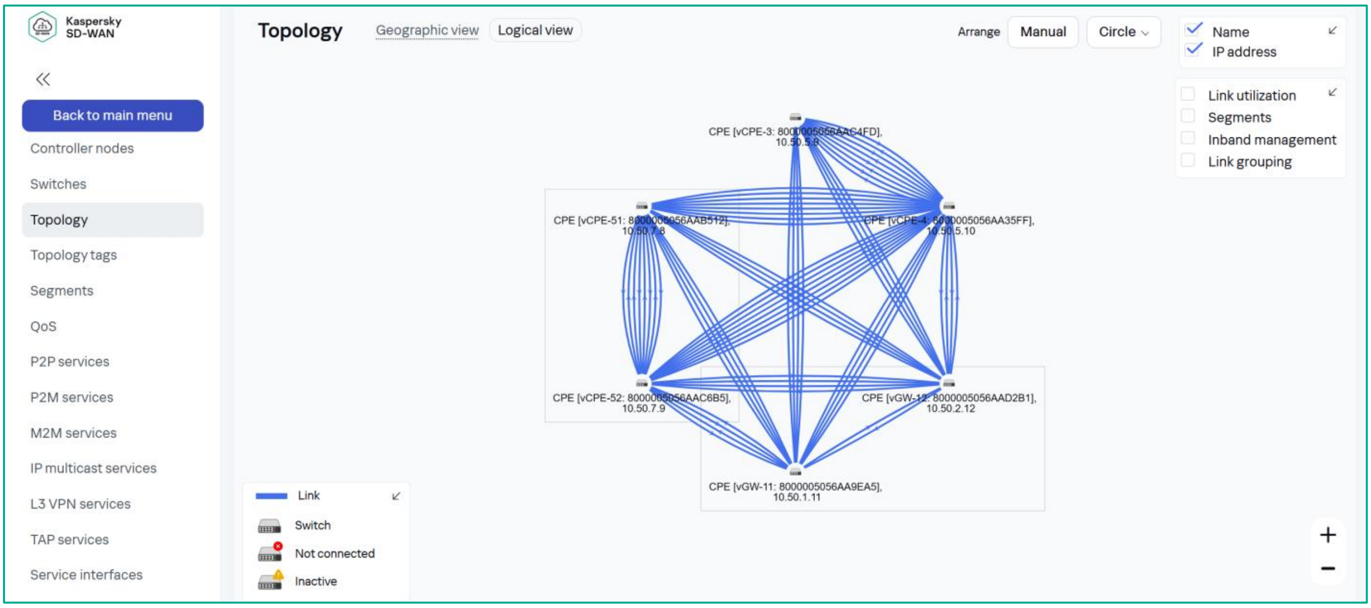
kaspersky

4.2.2. Display the built topology.

To view the built topology, go to **Infrastructure → SD-WAN cluster → Configuration menu → Topology**.

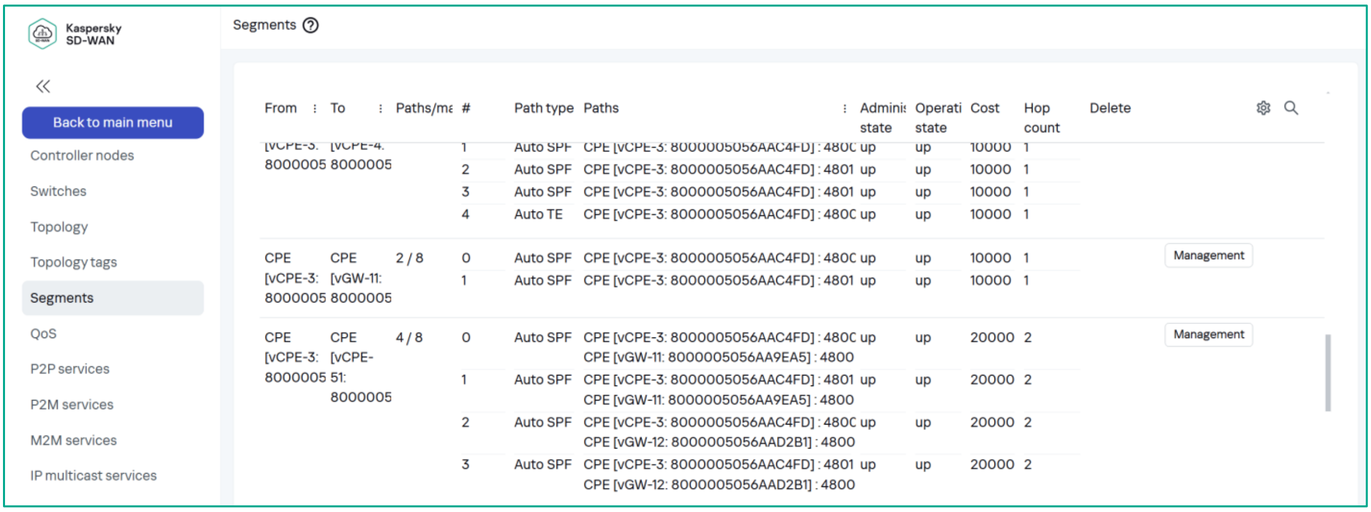
Open **Logical view** tab. Select **Arrange**: **Circle** for a more convenient view.

The screenshot shows that the CPE devices have established between links CPE-3 and CPE-4, Full-Mesh between CPE-4, CPE-51 and CPE-52, and have also retained links to the vGW-11/12 (gateways).



To check the segments between CPE devices, go to the **Segments** tab.

You see a list of segments showing the paths created between CPE devices. You can see that the new links form a partial mesh topology according to the configured tags (new links are created between **vCPE-4** and **vCPE-51/52**, but not between vCPE-3 and vCPE-51/52).



4.2.3. Restore the settings after the test is completed.

Remove tags from CPE devices added in 4.2.1.

kaspersky

4.3. **Creating topologies with transit CPEs**

The CPE supports a transit role for allowing building segments for adjacent CPE devices through the transit CPE. In this scenario, the partial mesh topology is used to demonstrate the functionality of transit CPEs.

Two groups of CPE devices are created:

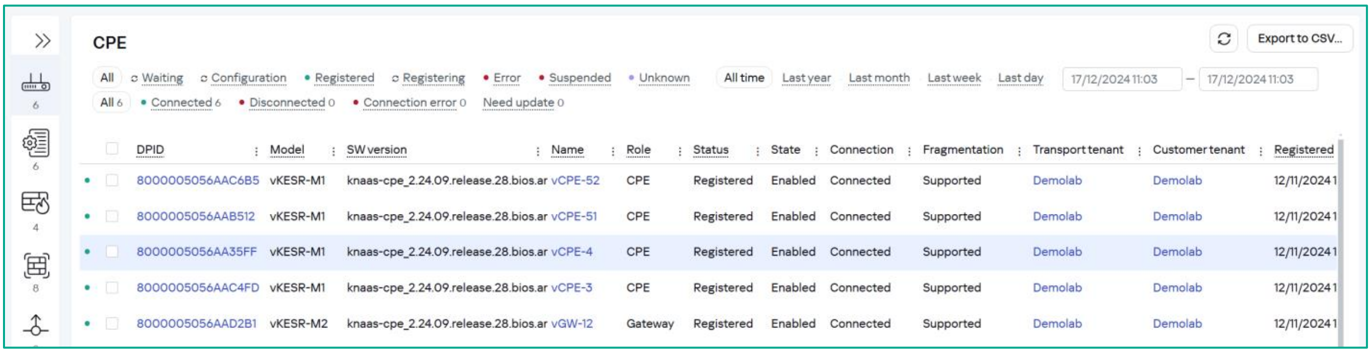
- vCPE-3 and vCPE-4.
- vCPE-4, vCPE-51, vCPE-52.

Each CPE device group is assigned its own topology tag. The vCPE-4 is assigned a transit role, allowing other CPEs to send traffic through this device.

4.3.1. Assign Topology Tags to the CPE devices.

To create a Partial-Mesh topology, you must assign different topology tags to the CPE devices according to the desired topology.

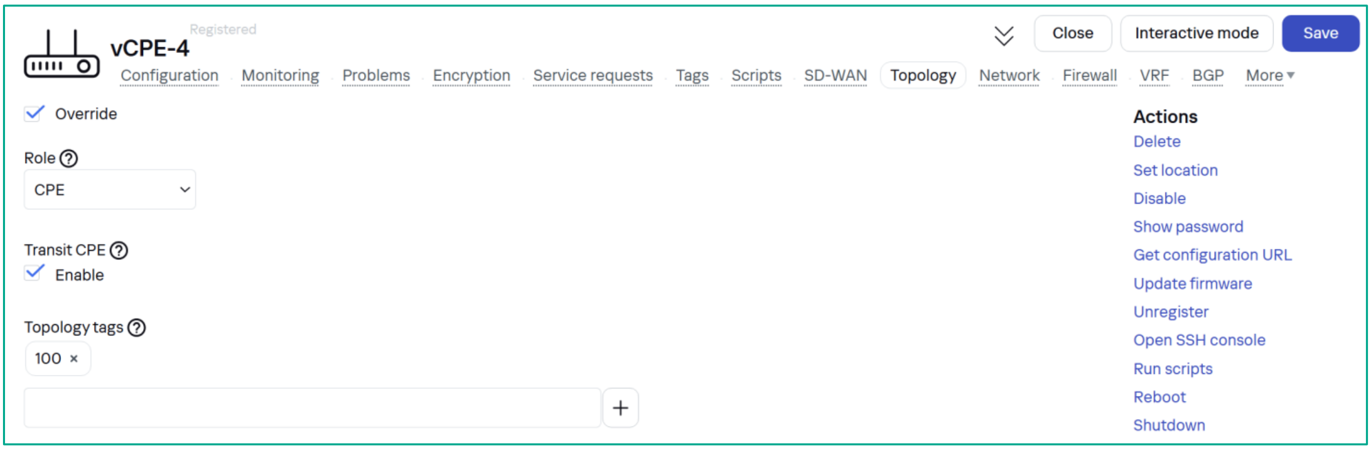
To configure them, go to the **CPE** menu and select **vCPE-3**.



Go to the **Topology** tab.

Set topologies parameters:

- Check **Override**.
- Check **Transit CPE**.
- Add tag **100** (click **+** to add).



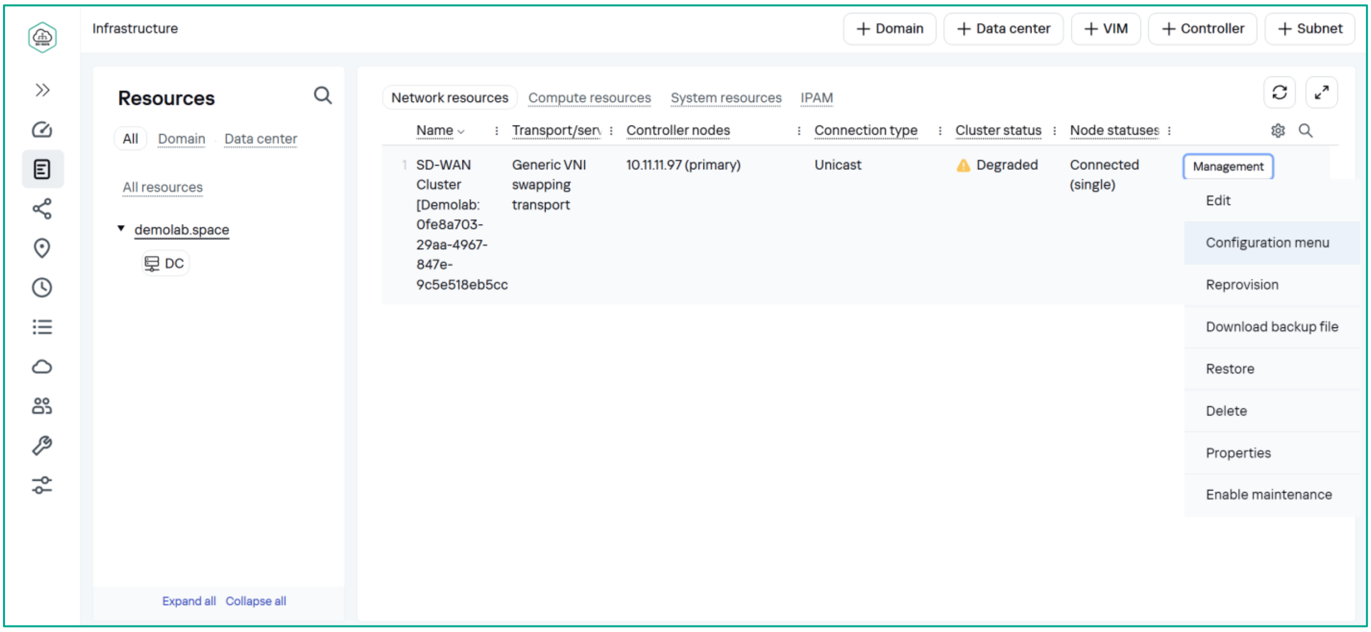
Click **Save** (the orchestrator will apply new settings to the CPE).

Assign topology tags to other CPE devices (these CPEs do not have a transit role and do not need to be configured as a transit CPE):

- **vCPE-51**: **200**.
- **vCPE-52**: **200**.
- **vCPE-4**: **100** and **200**.

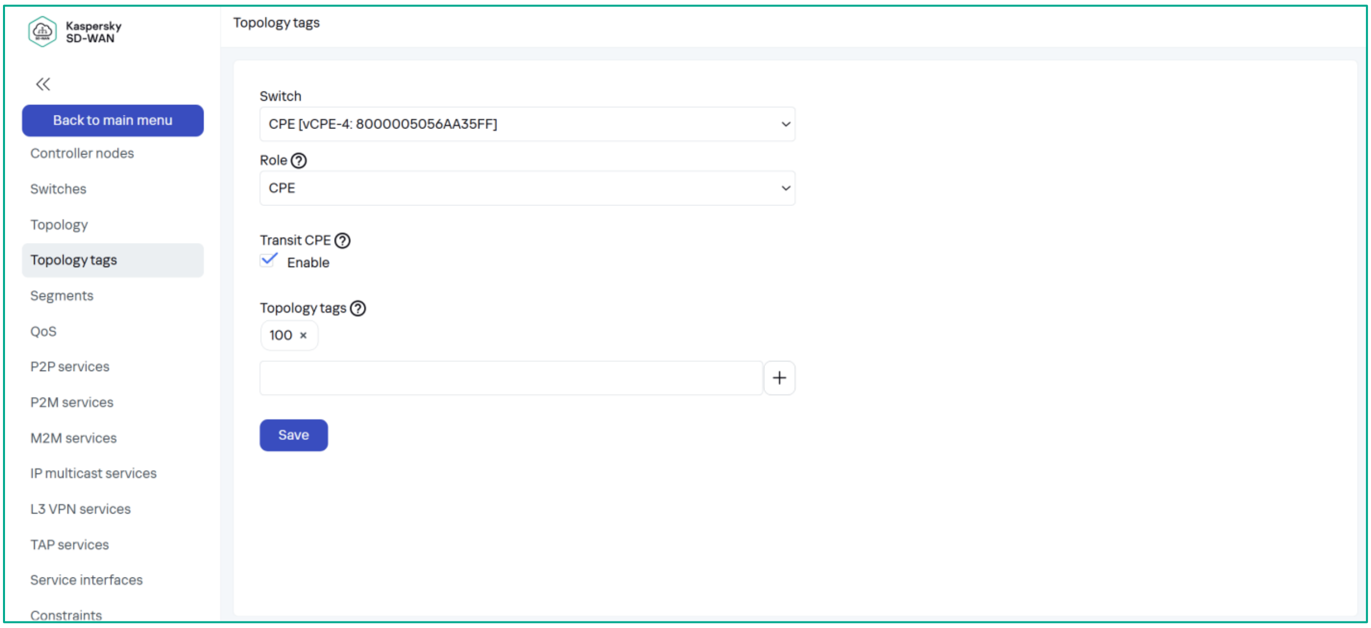
Alternatively, you can assign topology tags in the controller settings.

Go to **Infrastructure → SD-WAN cluster → Configuration menu**.



Open **Topology tags** menu**.**

Select **CPE** in the **Switch** selector and then add a topology tags (click **+**) or check **Transit CPE** parameter, then click **Save** to apply new settings.



4.3.2. Set maximum number of automatic SPF paths.

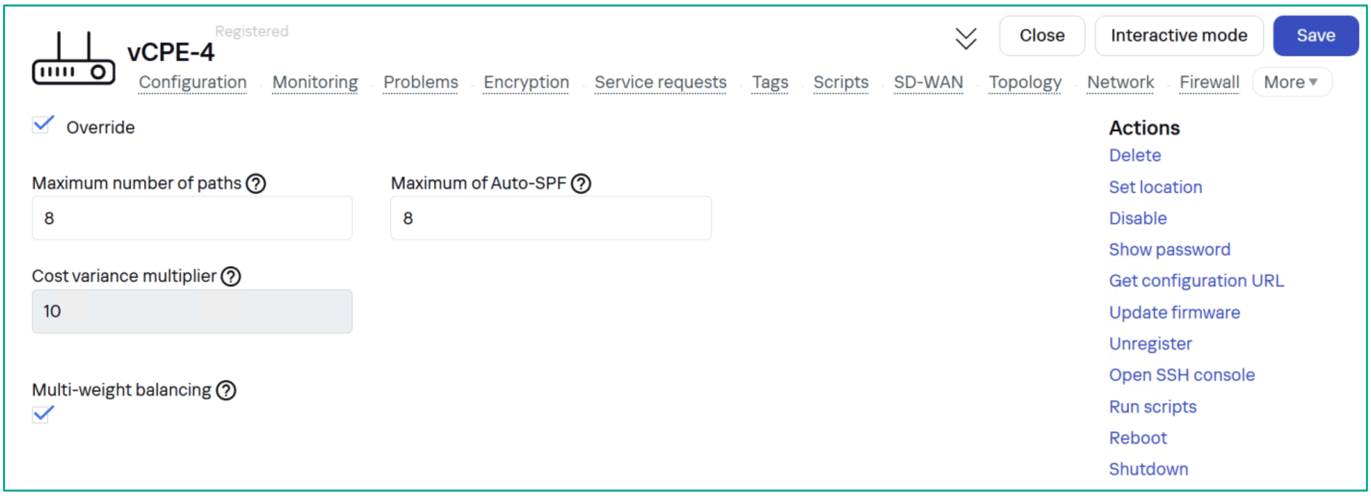
The scenario also requires increasing the maximum number of automatic SPF paths (default value is 2) to calculate additional segments through the transit CPE.

Go to the **CPE** menu and select **vCPE-4**.

Switch to the **Multipathing** tab.

Set SPF calculation parameters:

- Check **Override**.
- **Maximum of Auto-SPF**: **8**.



Click **Save**.

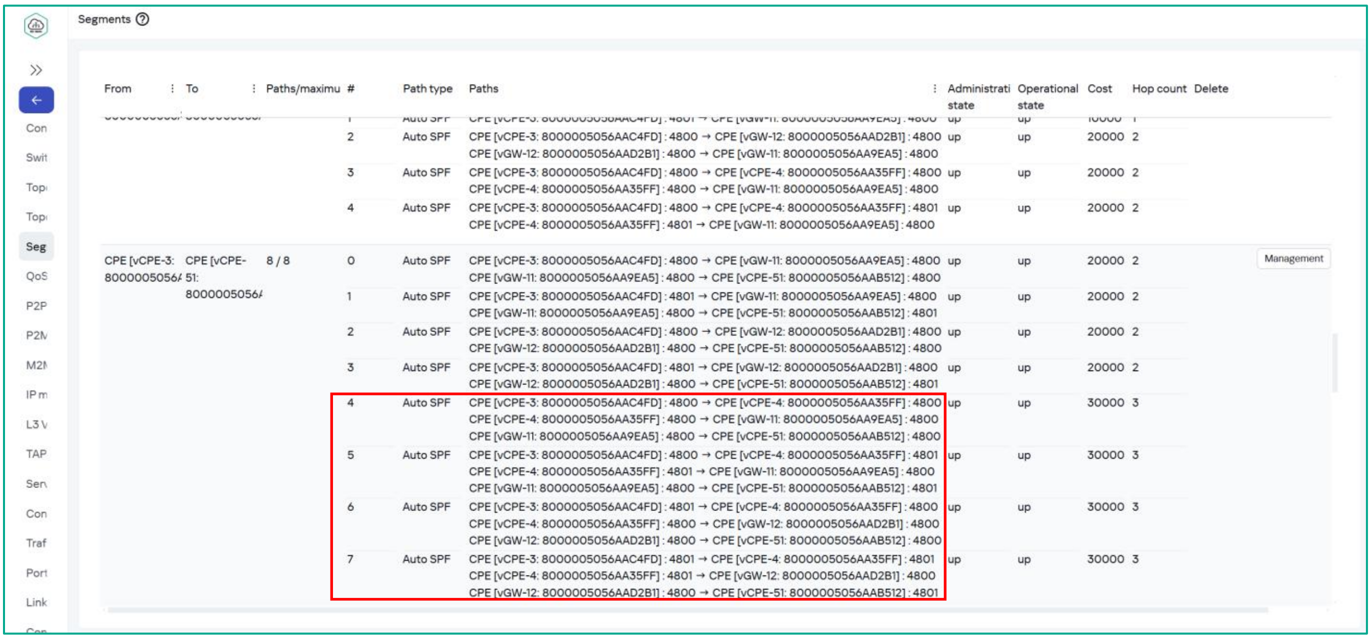
Repeat for **vCPE-3**, **vCPE-51** and **vCPE-52** devices.

4.3.3. Verify the created segments via vCPE-4.

To view segments, go to the **Infrastructure → SD-WAN cluster → Configuration menu → Segments**.

A list of segments will be displayed showing the calculated paths between CPE devices.

The screenshot shows the segment between **vCPE-3** and **vCPE-51**, the calculated paths include paths through **vCPE-4**, which is assigned the **Transit CPE** role.



4.3.4. Restore the settings after the test is completed.

Remove the topology tags and Auto-SPF values from the CPE devices added in 4.3.1 and 4.3.2.

kaspersky

Appendix A. PoC Checklist

Before running the tests, you should complete all the steps from the Kaspersky SD-WAN Proof of Concept Part 1 guide.

N	Test description	PoC item	Expected result	Actual result (pass / fail)
1	**Traffic management.**			
1.1	Load balancing with Active / Active link mode.	3.1	Traffic is balanced between the two WAN interfaces of the vCPE-3 device.	
1.2	Redundancy with Active/Standby link mode.	3.2	When the vCPE-3 device's primary WAN interface is operational, traffic does not flow through the backup WAN interface. When the primary WAN interface on the vCPE-3 device is disabled, traffic is switched to the backup WAN interface.	
1.3	Packet loss overcome with packet duplication in broadcast mode.	3.3	Packet copies from the vCPE-3 device are sent over the genev_sys_4800/4801 interfaces toward vGW-11/12.	
1.4	Network channels reliability with Forward Error Correction.	3.4	Enabling FEC reduces packet loss on the interface for which loss emulation is enabled.	
1.5	Enabling link packet loss monitoring.	3.4.2 - 3.4.3	When loss monitoring is enabled, the SD-WAN orchestrator displays packet loss statistics for links.	
1.6	Link quality monitoring (Jitter, Latency, Packet Loss).	3.5.1- 3.5.3	When latency and jitter monitoring is enabled, the SD-WAN orchestrator displays latency and jitter statistics for links.	
1.7	Manage traffic with Constraints.	3.5	When applying transport service constraints, links that do not meet the specified conditions are excluded from the traffic path (delay and jitter thresholds are set). In the iperf statistics, the jitter values for traffic passing from vCPE-3 to vCPE-4 are reduced.	
1.8	Classification of traffic using ACLs and redirecting it to links that meet the specified constraints.	3.6	Traffic that falls within the parameters of the created ACL (protocol UDP, port 5555) is redirected to links that are not marked as Last resort.	

N	Test description	PoC item	Expected result	Actual result (pass / fail)
1.9	Classification of traffic using DPI and redirecting it to links that meet the specified constraints.	3.7	DPI correctly matches traffic (SSH and HTTP). Test traffic is redirected to links that are not marked as "Last resort".	
2	**SD-WAN Topology Configuration.**			
2.1	Create Full-mesh topology.	4.1	After topology tags are configured, CPE devices create additional links to build a Full-Mesh topology (links are created from each CPE device to all other CPE devices).	
2.2	Create Partial-Mesh topology.	4.2	After configuring the topology tags, CPE devices create additional links to build the Partial-Mesh topology. There are 2 groups of CPEs: vCPE-3 and vCPE-4, and vCPE-51, vCPE-52, vCPE-4. The links of these groups build direct links to all devices in their group.	
2.3	Create topology with transit CPE.	4.3	The vCPE-3 and vCPE-51 devices create links through the vCPE-4 device marked as a transit.	

kaspersky

www.kaspersky.com
www.securelist.com