CRS3xx, CRS5xx, CCR2116, CCR2216 switch chip features

- **Summary**
  - Features
  - Models
  - Abbreviations
- **Port switching**
- **VLAN**
  - VLAN Filtering
    - Port-Based VLAN
    - MAC Based VLAN
    - Protocol Based VLAN
    - VLAN Tunneling (Q-in-Q)
  - Ingress VLAN translation
- **(R/M)STP**
- **Bonding**
- **Multi-chassis Link Aggregation Group**
- **L3 Hardware Offloading**
- **Port isolation**
- **IGMP/MLD Snooping**
- **DHCP Snooping and DHCP Option 82**
- **Controller Bridge and Port Extender**
- **Mirroring**
- **Traffic Shaping**
- **Traffic Storm Control**
- **MPLS hardware offloading**
- **Switch Rules (ACL)**
- **Port Security**
- **Dual Boot**
- **Configuring SwOS using RouterOS**
- **See also**

**Summary**

The CCR3xx, CRS5xx series switches and CCR2116, CCR2216 routers have highly integrated switches with high-performance CPU and feature-rich packet processors. These devices can be designed into various Ethernet applications including unmanaged switch, Layer 2 managed switch, carrier switch, inter-VLAN router, and wired unified packet processor.

⚠️ This article applies to CRS3xx, CRS5xx series switches, CCR2116, CCR2216 routers, and not to CRS1xx/CRS2xx series switches.

**Features**

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarding</td>
<td>- Configurable ports for switching or routing</td>
</tr>
<tr>
<td></td>
<td>- Full non-blocking wire-speed switching</td>
</tr>
<tr>
<td></td>
<td>- Large Unicast FDB for Layer 2 unicast forwarding</td>
</tr>
<tr>
<td></td>
<td>- Forwarding Databases works based on IVL</td>
</tr>
<tr>
<td></td>
<td>- Jumbo frame support</td>
</tr>
<tr>
<td></td>
<td>- IGMP Snooping support</td>
</tr>
<tr>
<td></td>
<td>- DHCP Snooping with Option 82</td>
</tr>
</tbody>
</table>
### Routing
- Layer 3 Hardware Offloading:
  - IPv4, IPv6 Unicast Routing
  - Supported on Ethernet, Bridge, Bonding, and VLAN interfaces
  - ECMP
  - Blackholes
  - Offloaded Fasttrack connections (applies only to certain switch models)
  - Offloaded NAT for Fasttrack connections (applies only to certain switch models)
  - Multiple MTU profiles

### Spanning Tree Protocol
- STP
- RSTP
- MSTP

### Mirroring
- Various types of mirroring:
  - Port based mirroring
  - VLAN based mirroring
  - MAC based mirroring

### VLAN
- Fully compatible with IEEE802.1Q and IEEE802.1ad VLAN
- 4k active VLANs
- Flexible VLAN assignment:
  - Port based VLAN
  - Protocol based VLAN
  - MAC based VLAN
- VLAN filtering
- Ingress VLAN translation

### Bonding
- Supports 802.3ad (LACP) and balance-xor modes
- Up to 8 member ports per bonding interface
- Hardware automatic failover and load balancing
- MLAG

### Traffic Shaping
- Ingress traffic limiting
  - Port based
  - MAC based
  - IP based
  - VLAN based
  - Protocol based
  - DSCP based
- Port based egress traffic limiting
- Traffic Storm Control

### Port isolation
- Applicable for Private VLAN implementation

### Access Control List
- Ingress ACL tables
- Classification based on ports, L2, L3, L4 protocol header fields
- ACL actions include filtering, forwarding and modifying of the protocol header fields

---

### Models

This table clarifies the main differences between Cloud Router Switch models and CCR routers.
<table>
<thead>
<tr>
<th>Model</th>
<th>Switch Chip</th>
<th>CPU</th>
<th>Cores</th>
<th>10G SFP+</th>
<th>2.5G Ethernet</th>
<th>10G Ethernet</th>
<th>25G SFP+</th>
<th>40G QSFP+</th>
<th>100G QSFP+</th>
<th>ACL rules</th>
<th>Unicast FDB entries</th>
<th>Jumbo Frame (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>netPower 15FR (CRS318-1F1-15Fr-2S)</td>
<td>Marvell-96DX2245</td>
<td>800MHz</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>16,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>netPower 16P (CRS318-16P-2S+)</td>
<td>Marvell-96DX2265</td>
<td>800MHz</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>16,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS310-1G-SS-4S+ (netFiber 9/IN)</td>
<td>Marvell-96DX2265</td>
<td>800MHz</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>16,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS310-8G-2S+</td>
<td>Marvell-96DX2265</td>
<td>800MHz</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>16,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS325-24G-2S+ (RM/IN)</td>
<td>Marvell-96DX3338</td>
<td>800MHz</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>16,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS328-24P-4S+</td>
<td>Marvell-96DX3338</td>
<td>800MHz</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>16,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS350-1G-4S+</td>
<td>Marvell-96DX3338</td>
<td>800MHz</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>16,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS309-1G-8S+</td>
<td>Marvell-96DX3328</td>
<td>800MHz</td>
<td>2</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1024</td>
<td>32,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS317-1G-16S+</td>
<td>Marvell-96DX3216</td>
<td>800MHz</td>
<td>2</td>
<td>16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1024</td>
<td>128,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS312-4C+3XG</td>
<td>Marvell-96DX3212</td>
<td>650MHz</td>
<td>1</td>
<td>4 (combo)</td>
<td>-</td>
<td>8 + 4 (combo)</td>
<td>-</td>
<td>-</td>
<td>512</td>
<td>32,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS326-24S+2Q+</td>
<td>Marvell-96DX3332</td>
<td>650MHz</td>
<td>1</td>
<td>24</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>256</td>
<td>32,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS354-48S+4X+2Q+</td>
<td>Marvell-96DX3257</td>
<td>650MHz</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>170</td>
<td>32,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS354-48P+4X+2Q+</td>
<td>Marvell-96DX3257</td>
<td>650MHz</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>170</td>
<td>32,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS504-4XG (IN/OUT)</td>
<td>Marvell-96DX3410</td>
<td>650MHz</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>1024</td>
<td>128,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS510-8XG-2XG-IN</td>
<td>Marvell-96DX3410</td>
<td>650MHz</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>1024</td>
<td>128,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CRS518-16XG-2XG</td>
<td>Marvell-96DX3525</td>
<td>650MHz</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>16</td>
<td>-</td>
<td>2</td>
<td>1024</td>
<td>128,000</td>
<td>10218</td>
<td></td>
</tr>
<tr>
<td>CCR2116-12G-4S+</td>
<td>Marvell-96DX3256</td>
<td>2000MHz</td>
<td>16</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>512</td>
<td>32,000</td>
<td>9570</td>
<td></td>
</tr>
<tr>
<td>CCR2216-1G-12XS-2XQ</td>
<td>Marvell-96DX3525</td>
<td>2000MHz</td>
<td>16</td>
<td>-</td>
<td>-</td>
<td>12</td>
<td>-</td>
<td>2</td>
<td>1024</td>
<td>128,000</td>
<td>9570</td>
<td></td>
</tr>
</tbody>
</table>

Abbreviations

- **FDB** - Forwarding Database
- **MDB** - Multicast Database
- **SVL** - Shared VLAN Learning
- **IVL** - Independent VLAN Learning
- **PVID** - Port VLAN ID
- **ACL** - Access Control List
- **CVID** - Customer VLAN ID
- **SVID** - Service VLAN ID

**Port switching**

In order to set up a port switching, check the Bridge Hardware Offloading page.

Currently, it is possible to create only one bridge with hardware offloading. Use the `hw=yes/no` parameter to select which bridge will use hardware offloading.
VLAN

Since RouterOS version 6.41, a bridge provides VLAN aware Layer2 forwarding and VLAN tag modifications within the bridge. This set of features makes bridge operation more like a traditional Ethernet switch and allows to overcome Spanning Tree compatibility issues compared to the configuration when tunnel-like VLAN interfaces are bridged. Bridge VLAN Filtering configuration is highly recommended to comply with STP (802.1D), RSTP (802.1w) standards and it is mandatory to enable MSTP (802.1s) support in RouterOS.

VLAN Filtering

VLAN filtering is described on the [Bridge VLAN Filtering](#) section.

VLAN setup examples

Below are describes some of the most common ways how to utilize VLAN forwarding.

**Port-Based VLAN**

The configuration is described on the [Bridge VLAN Filtering](#) section.

**MAC Based VLAN**

- The Switch Rule table is used for MAC Based VLAN functionality, see [this table](#) on how many rules each device supports.
- MAC-based VLANs will only work properly between switch ports and not between switch ports and CPU. When a packet is being forwarded to the CPU, the `pvid` property for the bridge port will be always used instead of `new-vlan-id` from ACL rules.
- MAC-based VLANs will not work for DHCP packets when DHCP snooping is enabled.

Enable switching on ports by creating a bridge with enabled hw-offloading:

```
/ interface bridge
add name=bridgel vlan-filtering=yes
/ interface bridge port
add bridge=bridgel interface=ether2 hw=yes
add bridge=bridgel interface=ether7 hw=yes
```

Add VLANs in the Bridge VLAN table and specify ports:

```
/ interface bridge vlan
add bridge=bridgel tagged=ether2 untagged=ether7 vlan-ids=200,300,400
```

Add Switch rules which assign VLAN id based on MAC address:

```
/ interface ethernet switch rule
```

**Protocol Based VLAN**
Enable switching on ports by creating a bridge with enabled hw-offloading:

```
//interface bridge
add name=bridge1 vlan-filtering=yes
//interface bridge port
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether6 hw=yes
add bridge=bridge1 interface=ether7 hw=yes
add bridge=bridge1 interface=ether8 hw=yes
```

Add VLANs in the Bridge VLAN table and specify ports:

```
//interface bridge vlan
add bridge=bridge1 tagged=ether2 untagged=ether6 vlan-ids=200
add bridge=bridge1 tagged=ether2 untagged=ether7 vlan-ids=300
add bridge=bridge1 tagged=ether2 untagged=ether8 vlan-ids=400
```

Add Switch rules which assign VLAN id based on MAC protocol:

```
//interface ethernet switch rule
add mac-protocol=ip new-vlan-id=200 ports=ether6 switch=switch1 vlan-id=10
add mac-protocol=ip new-vlan-id=300 ports=ether7 switch=switch1 vlan-id=20
```

**VLAN Tunneling (Q-in-Q)**

Since RouterOS v6.43 it is possible to use a provider bridge (IEEE 802.1ad) and Tag Stacking VLAN filtering, and hardware offloading at the same time. The configuration is described in the Bridge VLAN Tunneling (Q-in-Q) section.

```
//interface bridge
add name=bridge1 vlan-filtering=no
//interface bridge port
add interface=ether1 bridge=bridge1 hw=yes
add interface=ether2 bridge=bridge1 hw=yes
```

Ingress VLAN translation

It is possible to translate a certain VLAN ID to a different VLAN ID using ACL rules on an ingress port. In this example we create two ACL rules, allowing bidirectional communication. This can be done by doing the following.

Create a new bridge and add ports to it with hardware offloading:

```
//interface bridge
add name=bridge1 vlan-filtering=no
//interface bridge port
add interface=ether1 bridge=bridge1 hw=yes
add interface=ether2 bridge=bridge1 hw=yes
```

Add ACL rules to translate a VLAN ID in each direction:

```
//interface ethernet switch rule
add new-dst-ports=ether2 new-vlan-id=20 ports=ether1 switch=switch1 vlan-id=10
add new-dst-ports=ether1 new-vlan-id=10 ports=ether2 switch=switch1 vlan-id=20
```
Add both VLAN IDs to the bridge VLAN table:

```
/interface bridge vlan
add bridge=bridgel tagged=ether1 vlan-ids=10
add bridge=bridgel tagged=ether2 vlan-ids=20
```

Enable bridge VLAN filtering:

```
/interface bridge set bridgel vlan-filtering=yes
```

---

**Bidirectional communication is limited only between two switch ports. Translating VLAN ID between more ports can cause traffic flooding or incorrect forwarding between the same VLAN ports.**

**By enabling `vlan-filtering` you will be filtering out traffic destined to the CPU, before enabling VLAN filtering you should make sure that you set up a Management port.**

---

**R/M)STP**

CRS3xx, CRS5xx series switches, and CCR2116, CCR2216 routers are capable of running STP, RSTP, and MSTP on a hardware level. For more detailed information you should check out the [Spanning Tree Protocol](#) manual page.

**Bonding**

CRS3xx, CRS5xx series switches and CCR2116, CCR2216 routers support hardware offloading with bonding interfaces. Only 802.3ad and balance-xor bonding modes are hardware offloaded, other bonding modes will use the CPU's resources. You can find more information about the bonding interfaces in the [Bonding Interface](#) section. If 802.3ad mode is used, then LACP (Link Aggregation Control Protocol) is supported.

To create a hardware offloaded bonding interface, you must create a bonding interface with a supported bonding mode:

```
/interface bonding
add mode=802.3ad name=bond1 slaves=ether1,ether2
```

This interface can be added to a bridge alongside other interfaces:

```
/interface bridge
add name=bridge
/bridge port
add bridge=bridge interface=bond1 hw=yes
add bridge=bridge interface=ether3 hw=yes
add bridge=bridge interface=ether4 hw=yes
```

---

**Do not add interfaces to a bridge that are already in a bond, RouterOS will not allow you to add an interface to bridge that is already a slave port for bonding.**

Make sure that the bonding interface is hardware offloaded by checking the "H" flag:

```
/bridge port print
Flags: K - disabled, I - inactive, D - dynamic, H - hw-offload
# INTERFACE        BRIDGE  H
 0 bond1           bridge  yes
 1 ether3          bridge  yes
 2 ether4          bridge  yes
```
Multi-chassis Link Aggregation Group

MLAG (Multi-chassis Link Aggregation Group) implementation in RouterOS allows configuring LACP bonds on two separate devices, while the client device believes to be connected on the same machine. This provides a physical redundancy in case of switch failure. All CRS3xx, CRS5xx series and CCR2116, CCR2216 devices can be configured with MLAG. Read here for more information.

L3 Hardware Offloading

Layer3 hardware offloading (otherwise known as IP switching or HW routing) will allow to offload some of the router features onto the switch chip. This allows reaching wire speeds when routing packets, which simply would not be possible with the CPU.

Offloaded feature set depends on the used chipset. Read here for more info.

Port isolation

Since RouterOS v6.43 is it possible to create a Private VLAN setup, an example can be found in the Switch chip port isolation manual page. Hardware offloaded bonding interfaces are not included in the switch port-isolation menu, but it is still possible to configure port-isolation individually on each secondary interface of the bonding.

Port isolation can be used with vlan-filtering bridge and it is possible to isolate ports that are members of the same VLAN. The isolation works per-port, it is not possible to isolate ports per-VLAN.

IGMP/MLD Snooping

CRS3xx, CRS5xx series switches and CCR2116, CCR2216 routers are capable of using IGMP/MLD Snooping on a hardware level. To see more detailed information, you should check out the IGMP/MLD snooping manual page.

DHCP Snooping and DHCP Option 82

CRS3xx, CRS5xx series switches and CCR2116, CCR2216 routers are capable of using DHCP Snooping with Option 82 on a hardware level. The switch will create a dynamic ACL rule to capture the DHCP packets and redirect them to the main CPU for further processing. To see more detailed information, please visit the DHCP Snooping and DHCP Option 82 manual page.

DHCP snooping will not work when hardware offloading bonding interfaces are created.

Controller Bridge and Port Extender

Controller Bridge (CB) and Port Extender (PE) is an IEEE 802.1BR standard implementation in RouterOS. It allows virtually extending the CB ports with a PE device and managing these extended interfaces from a single controlling device. Such configuration provides a simplified network topology, flexibility, increased port density, and ease of manageability. See more details on Controller Bridge and Port Extender manual.

Mirroring
Mirroring lets the switch sniff all traffic that is going in a switch chip and send a copy of those packets out to another port (mirror-target). This feature can be used to easily set up a tap device that allows you to inspect the traffic on your network on a traffic analyzer device. It is possible to set up a simple port-based mirroring, but it is also possible to set up more complex mirroring based on various parameters. Note that mirror-target port has to belong to the same switch (see which port belongs to which switch in /interface ethernet menu). Also, mirror-target can have a special ‘cpu’ value, which means that sniffed packets will be sent out of switch chips CPU port. There are many possibilities that can be used to mirror certain traffic, below you can find the most common mirroring examples:

**Port Based Mirroring:**
```
/interface ethernet switch
set switch1 mirror-source=ether2 mirror-target=ether3
```

**Property** mirror-source will send an ingress and egress packet copies to the mirror-target port. Both mirror-source and mirror-target are limited to a single interface.

```
/interface ethernet switch
set switch1 mirror-source=none mirror-target=ether3
/interface ethernet switch rule
add mirror=yes ports=ether1,ether2 switch=switch1
```

**Using ACL rules, it is possible to mirror packets from multiple ports interfaces. Only ingress packets are mirrored to mirror-target interface.**

**VLAN Based Mirroring:**
```
/interface bridge
set bridge1 vlan-filtering=yes
/interface ethernet switch
set switch1 mirror-target=ether3 mirror-source=none
/interface ethernet switch rule
add mirror=yes ports=ether1 switch=switch1 vlan-id=11
```

**By enabling vlan-filtering you will be filtering out traffic destined to the CPU, before enabling VLAN filtering you should make sure that you set up a Management port.**

**MAC Based Mirroring:**
```
/interface ethernet switch
set switch1 mirror-target=ether3 mirror-source=none
/interface ethernet switch rule
```

**Protocol Based Mirroring:**
```
/interface ethernet switch
set switch1 mirror-target=ether3 mirror-source=none
/interface ethernet switch rule
add mirror=yes ports=ether1 switch=switch1 mac-protocol=ipx
```

**IP Based Mirroring:**
There are other options as well, check the ACL section to find out all possible parameters that can be used to match packets.

**Traffic Shaping**

It is possible to limit ingress traffic that matches certain parameters with ACL rules and it is possible to limit ingress/egress traffic per port basis. The policer is used for ingress traffic, the shaper is used for egress traffic. The ingress policer controls the received traffic with packet drops. Everything that exceeds the defined limit will get dropped. This can affect the TCP congestion control mechanism on end hosts and achieved bandwidth can be actually less than defined. The egress shaper tries to queue packets that exceed the limit instead of dropping them. Eventually, it will also drop packets when the output queue gets full, however, it should allow utilizing the defined throughput better.

**Port-based traffic police and shaper:**

```
/interface ethernet switch port
set ether1 ingress-rate=10M egress-rate=5M
```

**MAC-based traffic policer:**

```
/interface ethernet switch rule
```

**VLAN-based traffic policer:**

```
/interface bridge
set bridge1 vlan-filtering=yes
/interface ethernet switch rule
add ports=ether1 switch=switch1 vlan-id=11 rate=10M
```

By enabling `vlan-filtering` you will be filtering out traffic destined to the CPU, before enabling VLAN filtering you should make sure that you set up a Management port.

**Protocol-based traffic policer:**

```
/interface ethernet switch rule
add ports=ether1 switch=switch1 mac-protocol=ipx rate=10M
```

There are other options as well, check the ACL section to find out all possible parameters that can be used to match packets.

**Traffic Storm Control**
Since RouterOS v6.42 it is possible to enable traffic storm control. A traffic storm can emerge when certain frames are continuously flooded on the network. For example, if a network loop has been created and no loop avoidance mechanisms are used (e.g., Spanning Tree Protocol), broadcast or multicast frames can quickly overwhelm the network, causing degraded network performance or even complete network breakdown. With CRS3xx, CRS5xx series switches and CCR2116, CCR2216 routers it is possible to limit broadcast, unknown multicast and unknown unicast traffic. Unknown unicast traffic is considered when a switch does not contain a host entry for the destined MAC address. Unknown multicast traffic is considered when a switch does not contain a multicast group entry in the /interface bridge mdb menu. Storm control settings should be applied to ingress ports, the egress traffic will be limited.

The storm control parameter is specified in percentage (%) of the link speed. If your link speed is 1Gbps, then specifying storm-rate as 10 will allow only 100Mbps of broadcast, unknown multicast and/or unknown unicast traffic to be forwarded.

Sub-menu: /interface ethernet switch port

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit-broadcasts (yes / no; Default: yes)</td>
<td>Limit broadcast traffic on a switch port.</td>
</tr>
<tr>
<td>limit-unknown-multicasts (yes / no; Default: no)</td>
<td>Limit unknown multicast traffic on a switch port.</td>
</tr>
<tr>
<td>limit-unknown-unicasts (yes / no; Default: no)</td>
<td>Limit unknown unicast traffic on a switch port.</td>
</tr>
<tr>
<td>storm-rate (integer 0..100; Default: 100)</td>
<td>Amount of broadcast, unknown multicast and/or unknown unicast traffic is limited to in percentage of the link speed.</td>
</tr>
</tbody>
</table>

For example, to limit 1% (10Mbps) of broadcast and unknown unicast traffic on ether1 (1Gbps), use the following commands:

```
/interface ethernet switch port
set ether1 storm-rate=10 limit-broadcasts=yes limit-unknown-unicasts=yes
```

MPLS hardware offloading

Since RouterOS v6.41 it is possible to offload certain MPLS functions to the switch chip, the switch must be a (P)rovider router in a PE-P-PE setup in order to achieve hardware offloading. A setup example can be found in the Basic MPLS setup example manual page. The hardware offloading will only take place when LDP interfaces are configured as physical switch interfaces (e.g. Ethernet, SFP, SFP+).

Currently only CRS317-1G-16S+ and CRS309-1G-8S+ using RouterOS v6.41 and newer are capable of hardware offloading certain MPLS functions. CRS317-1G-16S+ and CRS309-1G-8S+ built-in switch chip is not capable of popping MPLS labels from packets, in a PE-P-PE setup you either have to use explicit null or disable TTL propagation in MPLS network to achieve hardware offloading.

The MPLS hardware offloading has been removed since RouterOS v7.

Switch Rules (ACL)

Access Control List contains ingress policy and egress policy engines. See this table on how many rules each device supports. It is an advanced tool for wire-speed packet filtering, forwarding and modifying based on Layer2, Layer3 and Layer4 protocol header field conditions.
ACL rules are checked for each received packet until a match has been found. If there are multiple rules that can match, then only the first rule will be triggered. A rule without any action parameters is a rule to accept the packet.

It is not required to set `mac-protocol` to certain IP version when using L3 or L4 matchers, however, it is recommended to set the `mac-protocol=ip` or `mac-protocol=ipv6` when filtering any IP packets.

When switch ACL rules are modified (e.g. added, removed, disabled, enabled, or moved), the existing switch rules will be inactive for a short time. This can cause some packet leakage during the ACL rule modifications.

### Sub-menu: /interface ethernet switch rule

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy-to-cpu (no</td>
<td>yes; Default: no)</td>
</tr>
<tr>
<td>disabled (yes</td>
<td>no; Default: no)</td>
</tr>
<tr>
<td>dscp (0..63)</td>
<td>Matching the DSCP field of the packet (only applies to IPv4 packets).</td>
</tr>
<tr>
<td>dst-address (IP address/Mask)</td>
<td>Matching destination IPv4 address and mask, also matches the destination IP in ARP packets.</td>
</tr>
<tr>
<td>dst-address6 (IPv6 address/Mask)</td>
<td>Matching destination IPv6 address and mask.</td>
</tr>
<tr>
<td>dst-mac-address (MAC address/Mask)</td>
<td>Matching destination MAC address and mask.</td>
</tr>
<tr>
<td>dst-port (0..65535)</td>
<td>Matching destination protocol port number (applies to IPv4 and IPv6 packets if <code>mac-protocol</code> is not specified).</td>
</tr>
<tr>
<td>flow-label (0..1048575)</td>
<td>Matching IPv6 flow label.</td>
</tr>
<tr>
<td>mac-protocol (802.2</td>
<td>arp</td>
</tr>
<tr>
<td>mirror (no</td>
<td>yes)</td>
</tr>
<tr>
<td>new-dst-ports (ports)</td>
<td>Changes the destination port as specified. An empty setting will drop the packet. A specified port will redirect the packet to it. When the parameter is not used, the packet will be accepted. Multiple &quot;new-dst-ports&quot; are not supported.</td>
</tr>
<tr>
<td>new-vlan-id (0..4095)</td>
<td>Changes the VLAN ID to the specified value. Requires <code>vlan-filtering=yes</code>.</td>
</tr>
<tr>
<td>new-vlan-priority (0..7)</td>
<td>Changes the VLAN priority (priority code point). Requires <code>vlan-filtering=yes</code>.</td>
</tr>
<tr>
<td>ports (ports)</td>
<td>Matching ports on which will the rule apply on received traffic.</td>
</tr>
<tr>
<td>protocol (dccp</td>
<td>ddp</td>
</tr>
<tr>
<td>rate (0.4294967295)</td>
<td>Sets ingress traffic limitation (bits per second) for matched traffic.</td>
</tr>
<tr>
<td>redirected-to-cpu (no</td>
<td>yes)</td>
</tr>
<tr>
<td>src-address (IP address/Mask)</td>
<td>Matching source IPv4 address and mask, also matches the source IP in ARP packets.</td>
</tr>
<tr>
<td><strong>src-address6 (IPv6 address/Mask)</strong></td>
<td>Matching source IPv6 address and mask.</td>
</tr>
<tr>
<td><strong>src-mac-address (MAC address/Mask)</strong></td>
<td>Matching source MAC address and mask.</td>
</tr>
<tr>
<td><strong>src-port (0..65535)</strong></td>
<td>Matching source protocol port number (applies to IPv4 and IPv6 packets if <code>mac-protocol</code> is not specified).</td>
</tr>
<tr>
<td><strong>switch (switch group)</strong></td>
<td>Matching switch group on which will the rule apply.</td>
</tr>
<tr>
<td><strong>traffic-class (0..255)</strong></td>
<td>Matching IPv6 traffic class.</td>
</tr>
<tr>
<td><strong>vlan-id (0..4095)</strong></td>
<td>Matching VLAN ID. Requires <code>vlan-filtering=yes</code>.</td>
</tr>
<tr>
<td>**vlan-header (not-present</td>
<td>present)**</td>
</tr>
<tr>
<td><strong>vlan-priority (0..7)</strong></td>
<td>Matching VLAN priority (priority code point).</td>
</tr>
</tbody>
</table>

**Action parameters:**

- copy-to-cpu
- redirect-to-cpu
- mirror
- new-dst-ports (can be used to drop packets)
- new-vlan-id
- new-vlan-priority
- rate

**Layer2 condition parameters:**

- dst-mac-address
- mac-protocol
- src-mac-address
- vlan-id
- vlan-header
- vlan-priority

**Layer3 condition parameters:**

- dscp
- protocol
- IPv4 conditions:
  - dst-address
  - src-address
- IPv6 conditions:
  - dst-address6
  - flow-label
  - src-address6
  - traffic-class

**Layer4 condition parameters:**

- dst-port
- src-port

---

**Warning:** For VLAN related matchers or VLAN related action parameters to work, you need to enable `vlan-filtering` on the bridge interface and make sure that hardware offloading is enabled on those ports, otherwise, these parameters will not have any effect.

---

**Warning:** When bridge interface `ether-type` is set to 0x8100, then VLAN related ACL rules are relevant to frames tagged using regular/customer VLAN (TPID 0x8100), this includes `vlan-id` and `new-vlan-id`. When bridge interface `ether-type` is set to 0x88a8, then ACL rules are relevant to frames tagged with 802.1ad service tag (TPID 0x88a8).
Port Security

It is possible to limit allowed MAC addresses on a single switch port. For example, to allow 64:D1:54:81:EF:8E MAC address on a switch port, start by switching multiple ports together, in this example 64:D1:54:81:EF:8E is going to be located behind ether1.

Create an ACL rule to allow the given MAC address and drop all other traffic on ether1 (for ingress traffic):

```
/interface ethernet switch rule
add new-dst-ports="" ports=ether1 switch=switch1
```

Switch all required ports together, disable MAC learning and disable unknown unicast flooding on ether1:

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 hw=yes learn=no unknown-unicast-flood=no
add bridge=bridge1 interface=ether2 hw=yes
```

Add a static hosts entry for 64:D1:54:81:EF:8E (for egress traffic):

```
/interface bridge host
add bridge=bridge1 interface=ether1 mac-address=64:D1:54:81:EF:8E
```

Broadcast traffic will still be sent out from ether1. To limit broadcast traffic flood on a bridge port, you can use the broadcast-flood parameter to toggle it. Do note that some protocols depend on broadcast traffic, such as streaming protocols and DHCP.

Dual Boot

The “dual boot” feature allows you to choose which operating system you prefer to use on CRS3xx series switches, RouterOS or SwOS. Device operating system could be changed using:

- Command-line (/system routerboard settings set boot-os=swos)
- Winbox
- Webfig
- Serial Console

More details about SwOS are described here: SwOS manual

Configuring SwOS using RouterOS

Since RouterOS 6.43 it is possible to load, save and reset SwOS configuration, as well as upgrade SwOS and set an IP address for the CRS3xx series switches by using RouterOS.

- Save configuration with /system swos save-config
- Load configuration with /system swos load-config
- Change password with /system swos password
- Reset configuration with /system swos reset-config
- Upgrade SwOS from RouterOS using /system swos upgrade

The configuration will be saved on the same device with swos.config as a filename, make sure you download the file from your device since the configuration file will be removed after a reboot.
The upgrade command will automatically install the latest available SwOS primary backup version, make sure that your device has access to the Internet in order for the upgrade process to work properly. When the device is booted into SwOS, the version number will include the letter "p", indicating a primary backup version. You can then install the latest available SwOS secondary main version from the SwOS "Upgrade" menu.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-acquisition-mode</td>
<td>Changes address acquisition method:</td>
</tr>
<tr>
<td>(d</td>
<td>dhcp-only - uses only a DHCP client to acquire address</td>
</tr>
<tr>
<td>hcp-only</td>
<td>dhcp-with-fallback - for the first 10 seconds will try to acquire address using a DHCP client. If the request is unsuccessful, then address falls back to static as defined by static-ip-address property</td>
</tr>
<tr>
<td>static)</td>
<td>static - address is set as defined by static-ip-address property</td>
</tr>
<tr>
<td>allow-from (IP/Mask; Default: 0.0.0.0/0)</td>
<td>IP address or a network from which the switch is accessible. By default, the switch is accessible by any IP address.</td>
</tr>
<tr>
<td>allow-from-ports (name; Default: )</td>
<td>List of switch ports from which the device is accessible. By default, all ports are allowed to access the switch</td>
</tr>
<tr>
<td>allow-from-vlan (integer: 0..4094; Default: 0)</td>
<td>VLAN ID from which the device is accessible. By default, all VLANs are allowed</td>
</tr>
<tr>
<td>identity (name; Default: Mikrotik)</td>
<td>Name of the switch (used for Mikrotik Neighbor Discovery protocol)</td>
</tr>
<tr>
<td>static-ip-address (IP; Default: 192.168.88.1)</td>
<td>IP address of the switch in case address-acquisition-mode is either set to dhcp-with-fallback or static. By setting a static IP address, the address acquisition process does not change, which is DHCP with fallback by default. This means that the configured static IP address will become active only when there is going to be no DHCP servers in the same broadcast domain</td>
</tr>
</tbody>
</table>

See also

- CRS Router
- CRS3xx VLANs with Bonds
- Basic VLAN switching
- Bridge Hardware Offloading
- Route Hardware Offloading
- Spanning Tree Protocol
- MTU on RouterBOARD
- Layer2 misconfiguration
- Bridge VLAN Table
- Bridge IGMP/MLD snooping
- Multi-chassis Link Aggregation Group