More advanced firewall setups, or complicated tasks, such as traffic prioritization, routing policies, where it is necessary to utilize more than one RouterOS facilities work together? What happens when and why? RouterOS packet flow diagram and flow examples will try to answers these questions.

It would be very complicated to represent what is going on with the packet in one diagram, therefore a packet flow diagram is divided into three parts:

- overall diagram;
- detailed bridging, routing, and MPLS flow diagram;
- diagram that shows what facilities and in what order is included in prerouting, input, forward, output and postrouting.

**Overall Packetflow Diagram**

Let's look at the overall diagram. It looks complicated at first, but after we go through the diagram with examples it will become much clearer.

There are 4 boxes in the center of the diagram: Bridging, Routing, Mpls decisions, and local router processes. So for example, if the packet needs to be routed or illustrated in the image below. Without looking deeper into each facility, packet enters in-interface, the router determines that it is IP traffic and needs to be routed processes and exits out-interface.

A simple explanation of each box before we go further with examples:

- **physical in-interface** - the starting point of the packet received by the router;
- **logical in-interface** - the starting point of the decapsulated packet (from tunnels, IPsec, etc);
- **local in** - the last point of packet destined to router itself;
- **interface HTB (Hierarchical Token Bucket)** - interface queue;
- **physical out-interface** - last point of the packet before it is actually sent out;
- **logical out-interface** - last point of the packet before encapsulation (to tunnels, IPsec, etc);
- **local out** - the starting point of a packet generated by the router;

Now it is time to take a deeper look at what is happening inside bridging, MPLS, and routing flows.
A simple explanation of each box before we go further with examples:

- **routing decision** - go through routes in the routing table to find a match for the destination IP address of the packet. When match is found - packet will be processed further, in case of no match - packet will be discarded;
- **mpls decision** - what to do with the packet based on MPLS forwarding tables;
- **bridging decision** - bridge goes through the MAC address table to find a match for destination MAC address of the packet. When a match is found - packet will be processed further, in case of no match - packets will be flooded (sent out via all bridge ports). A single packet copy will also reach a bridge input chain one of the many destinations;
- **use-ip-firewall** - whether a use-ip-firewall option is enabled in bridge settings;
- **ipsec-policy** - whether packet matches any of configured IPsec policies;

### Chains

RouterOS consist of a few default chains. These chains allow you to filter packets at various points:

- The **PREROUTING** chain: Rules in this chain apply to packets as they just arrive on the network interface. This chain is present in the nat, mangle and raw table.
- The **INPUT** chain: Rules in this chain apply to packets just before they're given to a local process. This chain is present in the mangle and filter tables.
- The **OUTPUT** chain: The rules here apply to packets just after they've been produced by a process. This chain is present in the raw, mangle, nat and filter tables.
- The **FORWARD** chain: The rules here apply to any packets that are routed through the current host. This chain is only present in the mangle and filter tables.
- The **POSTROUTING** chain: The rules in this chain apply to packets as they just leave the network interface. This chain is present in the nat and mangle tables.

Each of prerouting, input, forward, output and postrouting blocks contain even more facilities, which are illustrated in the third part of the packet flow diagram:

---

**Flow of Routed Packet**

**Forward**

Now, let's take our first example where packet gets routed over the router and look deeper through what facilities packet goes:

1. Packet enters prerouting processing:
   a. check if there is a hotspot and modify the packet for hotspot use
   b. process packet through RAW prerouting chain;
   c. send packet through connection tracking;
   d. process packet through Mangle prerouting chain;
   e. process packet through NATs dst-nat chain;
2. Run packet through routing table to make routing decision;
3. Packet enters postforward process:
   a. check TTL value;
b. process packet through Mangle forward chain;
c. process packet through Filter forward chain;
d. send packet to accounting processes;

4. A packet enters postrouting process;
   a. process packet through Mangle postrouting chain;
   b. process packet through NATs src-nat chain;
   c. if there is a hotspot undo any modifications made in hotspot-in;
   d. process packet through queue tree (HTB Global);
   e. process packet through simple queues;

5. Check if there is IPsec and process through IPsec policies;

Input
We already learned that packet goes into interface, the router determines that it is an IP packet and need to be routed, and here starts the complicated process:
1. A very similar process happens when a packet's destination is a router (routing input):

Packet enters prerouting processing:
   a. check if there is a hotspot and modify the packet for hotspot use;
   b. process packet through RAW prerouting chain;
   c. send packet through connection tracking;
   d. process packet through Mangle prerouting chain;
   e. process packet through NATs dst-nat chain;

2. Run packet through routing table to make routing decision;

3. A packet enters forward process:
   a. check TTL value;
   b. process packet through Mangle forward chain;
   c. process packet through Filter forward chain;
   d. send packet to accounting processes;

4. A packet enters postrouting process:
   a. process packet through Mangle postrouting chain;
   b. process packet through NATs src-nat chain;
   c. if there is a hotspot undo any modifications made in hotspot-in;
   d. process packet through queue tree (HTB Global);
   e. process packet through simple queues;

5. Check if there is IPsec and process through IPsec policies;

Output

Or when a packet is originated from the router (routing output):

1. The packet is originated from the router itself
   a. packet goes through the routing table to make a routing decision

2. A packet enters output process
   a. process packet through the Bridge decision;
   b. send packet through connection tracking;
   c. process packet through Mangle output chain;
   d. process packet through Filter output chain;
   e. send packet to routing adjustment (policy routing)

3. A packet enters postrouting process:
   a. process packet through Mangle postrouting chain;
   b. process packet through NATs src-nat chain;
   c. if there is a hotspot undo any modifications made in hotspot-in;
   d. process packet through queue tree (HTB Global);
   e. process packet through simple queues;

4. Check if there is IPsec and process through IPsec policies;
1. A packet enters the system.

2. Checks whether the use-ip-firewall option is enabled in the bridge settings.

3. Run packet through the bridge host table to make a forwarding decision. A packet that ends up being flooded (e.g. broadcast, multicast, unknown unicast traffic), gets multiplied per bridge port and then processed further in the bridge forward chain.
4. A packet goes through the bridge filter forward chain, where priority can be changed or packet can be simply accepted, dropped or marked;

5. Checks whether the use-ip-firewall option is enabled in the bridge settings;

6. A packet goes through the bridge NAT src-nat chain, where MAC source and priority can be changed, apart from that, a packet can be simply accepted, dropped or marked;

7. Checks whether the use-ip-firewall option is enabled in the bridge settings;
Bridge Input

Bridge input is a process that takes place when a packet is destined to the bridge interface. Most commonly this happens when you need to reach some service interface (e.g., a DHCP server) or you need to route traffic to other networks. The very first steps are similar to the bridge forward process - after receiving a packet on a bridge input interface, the device determines that the interface is a bridge port, so it gets passed through the bridging process:

1. A packet goes through the bridge NAT dst-nat chain, where MAC destination and priority can be changed, apart from that, a packet can be simply accepted, dropped or marked;
2. Checks whether the use-ip-firewall option is enabled in the bridge settings;
3. Run packet through the bridge host table to make a forwarding decision. A packet where destination MAC address matches with bridge MAC address will be passed to the bridge input chain. A packet that ends up being flooded (e.g., broadcast, multicast, unknown unicast traffic), also reaches bridge input chain as the bridge interface itself is one of the many destinations;
4. A packet goes through the bridge filter input chain, where priority can be changed or packet can be simply accepted, dropped or marked;
Bridge Output

Bridge output is a process that takes place when a packet should exit the device through one or multiple bridge ports. Most commonly this happens when a bridge device is connected to a certain bridge port (e.g., when a DHCP server running on a bridge interface is responding to a DHCP client). After a packet is processed by other processes and the device finally determines that the output interface is a bridge, the packet gets passed through the bridging process:

1. Run packet through the bridge host table to make a forwarding decision. A packet that ends up being flooded (e.g., broadcast, multicast, unknown unicast traffic), gets multiplied per bridge port and then processed further in the bridge output chain.
2. A packet goes through the bridge filter output chain, where priority can be changed or packet can be simply accepted, dropped or marked;
3. A packet goes through the bridge NAT src-nat chain, where MAC source and priority can be changed, apart from that, a packet can be simply accepted, dropped or marked;
4. Checks whether the use-ip-firewall option is enabled in the bridge settings;
Forward With Firewall Enabled

In certain network configurations, you might need to enable additional processing on routing chains for bridged traffic, for example, to use simple queues or IP firewall. If firewall is enabled under the bridge settings. Note that additional processing will consume more CPU resources to handle these packets. All the steps were all below is a recap:

1. A packet goes through the bridge NAT dst-nat chain;
2. With the use-ip-firewall option enabled, the packet will be further processed in the prerouting chain;
3. A packet enters prerouting processing;
4. Run packet through the bridge host table to make forwarding decision;
5. A packet goes through the bridge filter forward chain;
6. With the use-ip-firewall option enabled, the packet will be further processed in routing forward chain;
7. A packet enters routing forward processing;
8. A packet goes through the bridge NAT src-nat chain;
9. With the use-ip-firewall option enabled, the packet will be further processed in the postrouting chain;
10. A packet enters prerouting processing;
On the previous topic, we solely discussed a software bridging that requires the main CPU processing to forward packets through the correct bridge port. Most of the MikroTik devices are equipped with dedicated switching hardware, so-called switch chip or switch ASIC. This allows us to offload some of the bridging functions, like packet forwarding between bridge ports or packet filtering, to this specialized hardware chip without consuming any CPU resources. In RouterOS, we have named this function as Bridge Hardware (HW) Offloading. Different MikroTik devices might have different switch chips and each chip has a different set of features available, so make sure to visit this article to get more details - Bridge Hardware Offloading.
1. switching decision - widely depends on the switch model. This block controls all the switching related tasks, like host learning, packet forwarding, filtering, rate-limiting, VLAN tagging/untagging, mirroring, etc. Certain switch configuration can alter the packet flow;

2. switch-cpu port - a special purpose switch port for communication between the main CPU and other switch ports. Note that the switch-cpu port does not show up anywhere on RouterOS except for the switch menu, none of the software related configuration (e.g. interface-list) can be applied to this port. Packets that reach the CPU are automatically associated with the physical interface.

The hardware offloading, however, does not restrict a device to only hardware limited features, rather it is possible to take advantage of the hardware and software processing at the same time. This does require a profound understanding of how packets travel through switch chip and when exactly they are passed to the main CPU.

Switch Forward

We will further discuss a packet flow when a bridge hardware offloading is enabled and a packet is forwarded between two switched ports on a single switch chip. The simplest example:

1. The switch checks whether in-interface is hardware offloaded interface;
2. Run a packet through the switch host table to make a forwarding decision. If the switch finds a match for the destination MAC address, the packet is sent out.
through the physical interface. A packet that ends up being flooded (e.g. broadcast, multicast, unknown unicast traffic) gets multiplied and sent out to every hardware offloaded switch port.

**Switch to CPU Input**

This process takes place when a packet is received on a physical interface and it is destined to switch-cpu port for further software processing. There are two paths: hardware offloading and switching is not even used (e.g. a standalone interface for routing or a bridged interface but with deliberately disabled HW offloading), so for software processing. Another path is taken when hardware offloading is active on the in-interface. This will cause the packet to pass through the switching decision why switch might forward the packet to the switch-cpu port:

- a packet's destination MAC address match with a local MAC address, e.g. when a packet is destined to a local bridge interface;
- a switch might get flooded to all switch ports including the switch-cpu port, e.g. when broadcast, multicast or unknown unicast traffic is received;
- a switch might have learned that some hosts can only be reached through the CPU (switch-cpu port learning is discussed in the next section), e.g. when a bridged interface, such as wireless, EoIP and even Ethernet interfaces;
- a packet is intentionally copied to the switch-cpu, e.g. for a packet inspection;
- a packet is triggered by the switch configuration and should be processed in software, e.g. a DHCP or IGMP snooping.

See the packet walkthrough when an in-interface is hardware offloaded:

1. The switch checks whether in-interface is hardware offloaded interface;
2. Run a packet through the switch host table to make a forwarding decision. In case any of the above-mentioned points are true, the packet gets forwarded to the switch-cpu port.
3. The packet exits through the switch-cpu port and it will be further processed by the RouterOS packet flow.
CPU Output to Switch

This process takes place when a packet exits the RouterOS software processing and it is received on the switch-cpu port. Again, there are two paths the packet can take. One where hardware offloading and switching is not even used (e.g. a standalone interface for routing or a bridged interface but with deliberately disabled HW offloading), so the packet is simply sent out through the physical out-interface. Another path is taken when hardware offloading is active on the out-interface. This will cause the packet to pass through the switching decision. Just like any other switch port, the switch will learn the source MAC addresses from packets that are received on the switch-cpu port. This does come handy when a bridge contains HW and non-HW offloaded interfaces, so the switch can learn which frames should be forwarded to the CPU. See the packet walkthrough when an out-interface is hardware offloaded:

1. A packet that exits the RouterOS software processing is received on the switch-cpu port;
2. The switch checks whether out-interface is hardware offloaded interface;
3. Run a packet through the switch host table to make a forwarding decision. If the switch finds a match for the destination MAC address, the packet is sent out through the physical interface. A packet that ends up being flooded (e.g. broadcast, multicast, unknown unicast traffic) gets multiplied and sent out to every hardware offloaded switch port.

⚠ Any received packet that was flooded by the switch chip will not get flooded again by the software bridge to the same HW offloaded switch group. This prevents the formation of duplicate packets.
A software bridge that sends a flooded packet through HW offloaded interfaces, will only send a single packet copy per HW offloaded switch group rather The actual flooding will be done by the switch chip, this prevents the formation of duplicate packets.

Flow of MPLS Packet

1. Pop Label
   1.
   2.
   3.
Switch Label
1.
2.

Push Label
1.
2.
Logical Interfaces

So far we looked at examples when in or out interfaces are actual physical interfaces (Ethernet, wireless), but how packets will flow if the router receives tunnel encapsulated packets?

Let's assume that there is an IPIP packet coming in the router. Since it is a regular IPv4 packet it will be processed through all routing-related facilities (until "J" in the diagram). Then the router will look if packet needs to be decapsulated. In this case, it is an IPIP packet so "yes" send packet to decapsulation. After that packet will go another loop through facilities but this time as a decapsulated IPv4 packet.

It is very important because the packet actually travels through the firewall twice, so if there is a strict firewall, then there should accept rules for IPIP encapsulated IP packet.

IPSec Policies

Let's take a look at another tunnel type - IPSec. This type of VPN does not have logical interfaces but is processed in a similar manner. Instead of logical interfaces packets are processed through IPSec policies. After routing decision (2) and input firewall processing (3), the router tries to match the source and destination to IPSec policy. When policy matches the packet it is sent to decryption (5). After the decryption packet enters PREROUTING processing again (6) and starts another processing loop, but now with decapsulated packet.
The same process is with encapsulation but in reverse order. First IP packet gets processed through facilities, then matched against IPsec policies (5), encapsulated (6) and then sent to processing on the second loop (7-10).
From what we learned so far, it is quite obvious that such packet processing takes a lot of CPU resources. To fast things up FastPath was introduced since the first RouterOS v6. What it does is it skips processing in the Linux kernel, basically trade some RouterOS functionality for performance. For FastPath to work, interface driver support and specific configuration conditions are required.

### How Fast Path Works

FastPath is an interface driver extension, that allows a driver to talk directly to specific RouterOS facilities and skipping all others.

Packet can be forwarded by a fast path handler only if at least source interface supports a fast path. For complete fast-forwarding, destination interface support is also required.

Currently, RouterOS has following FastPath handlers:

- IPv4
- IPv4 FastTrack
- Traffic Generator
- MPLS
- Bridge

IPv4 FastPath handler is used if the following conditions are met:

- firewall rules are not configured;
- firewall address lists are not configured;
- simple and queue trees with parent=global are not configured;
- no mesh, metarouter interface configuration;
- sniffer, torch and traffic generator is not running;
- connection tracking is not active;
- ip accounting is disabled;
- VRFs are not configured (/ip route vrf is empty);
- Hotspot is not used (/ip hotspot has no interfaces);
- IPSec policies are not configured;
- /tool mac-scan is not actively used;
- /tool ip-scan is not actively used.

⚠️ Packets will travel FastPath way if FastTrack is used no matter if the above conditions are met.
Traffic Generator and MPLS automatically use FastPath if interface support this feature. Currently, MPLS fast-path applies only to MPLS switched traffic (frames if leave router as MPLS) - MPLS ingress and egress (including VPLS tunnel endpoints that do VPLS encap/decap) will operate as before.

Bridge handler is used if the following conditions are met:

- there are no bridge firewall rules;
- use-ip-firewall is disabled;
- no mesh, MetaRouter interface configuration;
- sniper, torch and traffic generator is not running.

Interfaces that support FastPath:

<table>
<thead>
<tr>
<th>RouterBoard</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB6xx series</td>
<td>ether1,2</td>
</tr>
<tr>
<td>RB7xx series</td>
<td>all ports</td>
</tr>
<tr>
<td>RB800</td>
<td>ether1,2</td>
</tr>
<tr>
<td>RB9xx series</td>
<td>all ports</td>
</tr>
<tr>
<td>RB1000</td>
<td>all ports</td>
</tr>
<tr>
<td>RB1100 series</td>
<td>ether1-11</td>
</tr>
<tr>
<td>RB2011 series</td>
<td>all ports</td>
</tr>
<tr>
<td>RB3011 series</td>
<td>all ports</td>
</tr>
<tr>
<td>CRS series routers</td>
<td>all ports</td>
</tr>
<tr>
<td>CCR series routers</td>
<td>all ports</td>
</tr>
<tr>
<td>All devices</td>
<td>wireless interfaces</td>
</tr>
<tr>
<td></td>
<td>bridge interfaces</td>
</tr>
<tr>
<td></td>
<td>vlan, vrrp interfaces</td>
</tr>
<tr>
<td></td>
<td>bonding interfaces (RX only)</td>
</tr>
<tr>
<td></td>
<td>PPPoe, L2TP interfaces</td>
</tr>
<tr>
<td></td>
<td>eoip, gre, ipip interfaces</td>
</tr>
</tbody>
</table>

EoIP, Gre, IPIP and L2TP interfaces have per-interface setting allow-fast-path. Allowing fast path on these interfaces have side effect of bypassing firewall, connect tree with parent=global, IP accounting, IPsec, hotspot universal client, vrf assignment for encapsulated packets that go through fast path. Also, packet fragments cannot be received in FastPath.

Only interface queue that guarantees FastPath is only-hardware-queue. If you need interface queue other than hardware then the packet will not go fully FastPath performance, as "interface queue" is the last step in the packet flow.

Packet may go Half-FastPath by switching from FastPath to SlowPath, but not the other way around. So, for example, if the receiving interface has FastPath support the router will process the packet by FastPath handlers as far as it can and then proceed with SlowPath. If the receiving interface does not support FastPath but processed by SlowPath all the way through the router.

FastPath support can be verified by checking fast-path property value in `/interface print detail`.

Fasttrack
Fasttrack can be decoded as Fast Path + Connection Tracking. It allows to mark connections as "fastracked", after marking packets that belong to fastracked connection table entry for such connection now will have fastracked flag.

Routing Forwarding FastPath

To mark a connection as fastracked new action was implemented fasttrack-connection for firewall filter and mangle. Currently, only IPv4 TCP and UDP conn maintain connection tracking entries some random packets will still be sent to slow path. This must be taken into consideration when designing firewalls with enabled FastTrack handler also supports source and destination NAT, so special exceptions for NATed connections are not required.

Traffic that belongs to fastracked connection travels in FastPath, which means that it will not be visible by other router L3 facilities (firewall, queues, IPsec, L etc).

Easiest way to start using this feature on home routers is to enable fasttrack for all established, related connections:

```
/ip firewall filter
add chain=forward action=fasttrack-connection connection-state=established,related \ comment="fasttrack established/related"
add chain=forward action=accept connection-state=established,related \ comment="accept established/related"
```

Notice that first rule marks established/related connections as fastracked, second rule is still required to accept packets belonging to those connections. The reason earlier, some random packets from fastracked connections are still sent the slow path way and only UDP and TCP are fastracked, but we still want to accept packets for other protocols.
After adding fasttrack rule special dummy rule appeared at the top of the list. This is not an actual rule, it is for visual information showing that some of the traffic is traveling FastPath and will not reach other firewall rules. These rules appear as soon as there is at least one "fasttracked" connection tracking entry and will disappear after the last "fasttracked" connection times out in the connection table.

The connection is FastTracked until a connection is closed, timed out or router is rebooted.

The following document in DOCX format describes the diagram in a way optimized for visually impaired people. The descriptions are by Apex CoVantage care of Benetech. They are not being updated.

- Packet flow, optimized document.