Netfilter's flowtable infrastructure

This documentation describes the Netfilter flowtable infrastructure which allows you to define a fastpath through the flowtable datapath. This infrastructure also provides hardware offload support. The flowtable supports for the layer 3 IPv4 and IPv6 and the layer 4 TCP and UDP protocols.

Overview

Once the first packet of the flow successfully goes through the IP forwarding path, from the second packet on, you might decide to offload the flow to the flowtable through your ruleset. The flowtable infrastructure provides a rule action that allows you to specify when to add a flow to the flowtable.

A packet that finds a matching entry in the flowtable (ie. flowtable hit) is transmitted to the output netdevice via neigh_xmit(), hence, packets bypass the classic IP forwarding path (the visible effect is that you do not see these packets from any of the Netfilter hooks coming after ingress). In case that there is no matching entry in the flowtable (ie. flowtable miss), the packet follows the classic IP forwarding path.

The flowtable uses a resizable hashtable. Lookups are based on the following n-tuple selectors: layer 2 protocol encapsulation (VLAN and PPPoE), layer 3 source and destination, layer 4 source and destination ports and the input interface (useful in case there are several conntrack zones in place).

The 'flow add' action allows you to populate the flowtable, the user selectively specifies what flows are placed into the flowtable. Hence, packets follow the classic IP forwarding path unless the user explicitly instruct flows to use this new alternative forwarding path via policy.

The flowtable datapath is represented in Fig.1, which describes the classic IP forwarding path including the Netfilter hooks and the flowtable fastpath bypass.

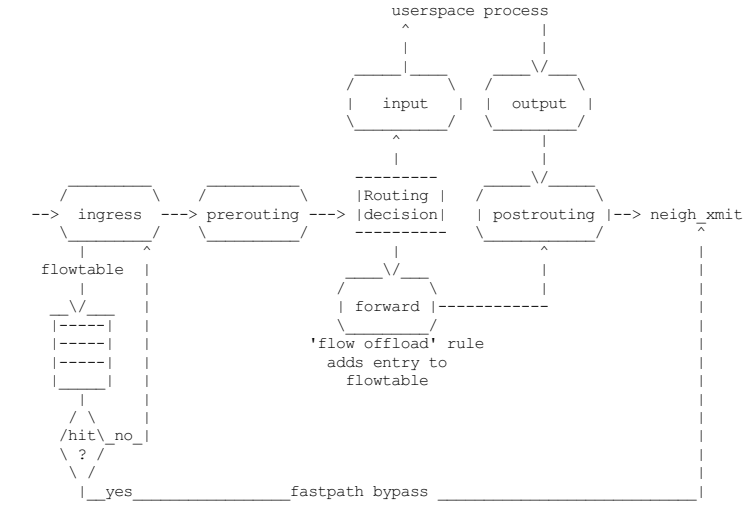


Fig.1 Netfilter hooks and flowtable interactions

The flowtable entry also stores the NAT configuration, so all packets are mangled according to the NAT policy that is specified from the classic IP forwarding path. The TTL is decremented before calling neigh_xmit(). Fragmented traffic is passed up to follow the classic IP forwarding path given that the transport header is missing, in this case, flowtable lookups are not possible. TCP RST and FIN packets are also passed up to the classic IP forwarding path to release the flow gracefully. Packets that exceed the MTU are also passed up to the classic forwarding path to report packet-too-big ICMP errors to the sender.

Example configuration

Enabling the flowtable bypass is relatively easy, you only need to create a flowtable and add one rule to your forward chain:

```
table inet x {
        flowtable f {
                hook ingress priority 0; devices = { eth0, eth1 };
        }
        chain y {
                type filter hook forward priority 0; policy accept;
                ip protocol tcp flow add @f
                counter packets 0 bytes 0
        }
}
```

This example adds the flowtable 'f' to the ingress hook of the eth0 and eth1 netdevices. You can create as many flowtables as you want in case you need to perform resource partitioning. The flowtable priority defines the order in which hooks are run in the pipeline, this is convenient in case you already have a nftables ingress chain (make sure the flowtable priority is smaller than the nftables ingress chain hence the flowtable runs before in the pipeline).

The 'flow offload' action from the forward chain 'y' adds an entry to the flowtable for the TCP syn-ack packet coming in the reply direction. Once the flow is offloaded, you will observe that the counter rule in the example above does not get updated for the packets that are being forwarded through the forwarding bypass.

You can identify offloaded flows through the [OFFLOAD] tag when listing your connection tracking table.

```
# conntrack -L
tcp      6 src=10.141.10.2 dst=192.168.10.2 sport=52728 dport=5201 src=192.168.10.2 dst=192.168.10.1 sport=5201 dport=52728
```

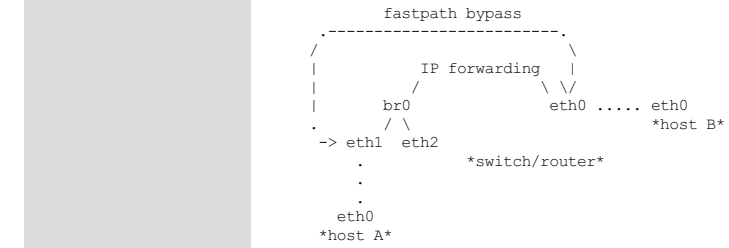
Layer 2 encapsulation

Since Linux kernel 5.13, the flowtable infrastructure discovers the real netdevice behind VLAN and PPPoE netdevices. The flowtable software datapath parses the VLAN and PPPoE layer 2 headers to extract the ethertype and the VLAN ID / PPPoE session ID which are used for the flowtable lookups. The flowtable datapath also deals with layer 2 decapsulation.

You do not need to add the PPPoE and the VLAN devices to your flowtable, instead the real device is sufficient for the flowtable to track your flows.

Bridge and IP forwarding

Since Linux kernel 5.13, you can add bridge ports to the flowtable. The flowtable infrastructure discovers the topology behind the bridge device. This allows the flowtable to define a fastpath bypass between the bridge ports (represented as eth1 and eth2 in the example figure below) and the gateway device (represented as eth0) in your switch/router.



The flowtable infrastructure also supports for bridge VLAN filtering actions such as PVID and untagged. You can also stack a classic VLAN device on top of your bridge port.

If you would like that your flowtable defines a fastpath between your bridge ports and your IP forwarding path, you have to add your bridge ports (as represented by the real netdevice) to your flowtable definition.

Counters

The flowtable can synchronize packet and byte counters with the existing connection tracking entry by specifying the counter statement in your flowtable definition, e.g.

```
table inet x {
        flowtable f {
                hook ingress priority 0; devices = { eth0, eth1 };
                counter
        }
}
```

Counter support is available since Linux kernel 5.7.

Hardware offload

If your network device provides hardware offload support, you can turn it on by means of the 'offload' flag in your flowtable definition, e.g.

```
table inet x {
        flowtable f {
                hook ingress priority 0; devices = { eth0, eth1 };
                flags offload;
        }
}
```

There is a workqueue that adds the flows to the hardware. Note that a few packets might still run over the flowtable software path until the workqueue has a chance to offload the flow to the network device.

You can identify hardware offloaded flows through the [HW_OFFLOAD] tag when listing your connection tracking table. Please, note that the [OFFLOAD] tag refers to the software offload mode, so there is a distinction between [OFFLOAD] which refers to the software flowtable fastpath and [HW_OFFLOAD] which refers to the hardware offload datapath being used by the flow.

The flowtable hardware offload infrastructure also supports for the DSA (Distributed Switch Architecture).

Limitations

The flowtable behaves like a cache. The flowtable entries might get stale if either the destination MAC address or the egress netdevice that is used for transmission changes.

This might be a problem if:

- You run the flowtable in software mode and you combine bridge and IP forwarding in your setup.
- Hardware offload is enabled.

More reading

This documentation is based on the LWN.net articles [1][2]. Rafal Milecki also made a very complete and comprehensive summary called "A state of network acceleration" that describes how things were before this infrastructure was mainlined [3] and it also makes a rough summary of this work [4].

[1] https://lwn.net/Articles/738214/

[2] https://lwn.net/Articles/742164/

[3] http://lists.infradead.org/pipermail/lede-dev/2018-January/010830.html

[4] http://lists.infradead.org/pipermail/lede-dev/2018-January/010829.html