Implementing Open vSwitch datapath using TC

Jiří Pírko
jiri@resnulli.us
Red Hat

What? Why?

- Goal:
  - Provide an alternative approach to the current OVS kernel datapath
  - Eventually remove OVS kernel datapath completely
- Motivation:
  - Increasing amount of features is implemented in multiple kernel parts
  - Prevent code duplication
  - Easier maintainability
  - Lower bug amount -> happier user
  - Hardware offload – do it in one place instead of two
  - It should have been done this way from the very beginning

OVS kernel datapath overview

- Match-action forwarding datapath
- Flow
  - Inherited from OpenFlow standard
  - Consists of:
    - Flow key – represents a set of fields in a packet to match
    - Flow mask – bitmask for flow key
    - Actions – set of various actions to be executed in case packet matches the key with mask
- Vports
  - Could be backed by a real netdevice (eth0)
  - Multiple vports are put into group called “bridge”
  - A bridge is represented by an “internal vport” backed by a netdevice
  - Tunnels (VXLAN, Geneve, GRE)
    - Not represented by netdevices

Classifier-Action subsystem of TC

- Present in kernel for ages (older than git kernel tree)
- Details explained in Jamal's talk/tutorial
- Consists of:
  - Classifiers
    - Do matching on packets
    - cls_32, cls_bpf, ...
  - Actions
    - Executed upon match
    - Chains
    - act_skbedit, act_nat, act_mirred, act_vlan, ...

Using TC Classifier-Action to implement OVS kernel DP

- TC CA and OVS both do match-action processing
- Why not to replace OVS by TC CA?
  - Many of needed TC CA features are already in place
  - Some of them are not
    - So implement them
- Classifiers attached to ingress qdisc

```
tc qdisc add dev eth0 ingress
tc filter add dev eth0 parent ffff: protocol all u32 match u32 0 0 \
        action mirred egress redirect dev eth1
tc qdisc add dev eth1 ingress
tc filter add dev eth1 parent ffff: protocol all u32 match u32 0 0 \
        action mirred egress redirect dev eth0
```

Ok, so what do we need?

The classifier

- cls_u32 could be able to cover
  - Some wrapper would be needed
  - Generic – probably has performance impacts
- cls_openflow – a new one
  - Implements matches on OpenFlow essential items
  - Planned to implement all items OVS uses

```
tc qdisc add dev eth0 ingress
tc filter add dev eth0 parent ffff: protocol all openflow \
                        src_ip 192.168.0.1 dst_ip 192.168.10.0/24 \
                        action mirred egress redirect dev eth1
```

Actions

- Output action
  - act_mirred
- Upstream output action
  - In case of a flow-miss
  - Forward to a tap device
- VLAN header pop and push actions
  - act_vlan
- MPLS
  - Similar to VLAN to some extend
  - Not implemented in TC CA yet

```
tc qdisc add dev eth0 ingress
tc filter add dev eth0 parent ffff: protocol all openflow \
                        src_ip 192.168.0.1 dst_ip 192.168.10.0/24 \
                        action mirred egress redirect dev eth1
```

Tunneling

- VXLAN, Geneve, GRE, ...
- In OVS tunnels are virtual ports without any netdevice
  - Open tunnel socket, register a rx function, on tx call tx function
- For TC, create a netdevice for tunnel and use it as any other netdevice
  - Not all tunnels available (Geneve)
  - Scalability issues (netdevice for thousands of tunnels)
- Tunnel action?
  - Action created socket and uses tx function for tx
  - Nowhere to hook on receive path
- Use “named” sockets
  - Create tunnel socket from userspace
  - Name it
  - Use in TC rules instead of netdevice

Named tunnel socket example

```
ip link add vxlansock0 type vxlan \
                    id 42 group 239.1.1.1 dev eth1 sock
tc qdisc add dev eth0 ingress
tc filter add dev eth0 parent ffff: protocol all \
                    u32 match u32 0 0 \
                    action mirred egress redirect sock vxlansk0
tc qdisc add sock vxlansock0 ingress
tc filter add sock vxlansock0 parent ffff: protocol all \
                    u32 match u32 0 0 \
                    action mirred egress redirect dev eth0
```

Other approaches

- nftables?
- eBPF?

The end

- Questions?