BCM5750X

Introduction to TruFlow

Application Note

Table of Contents

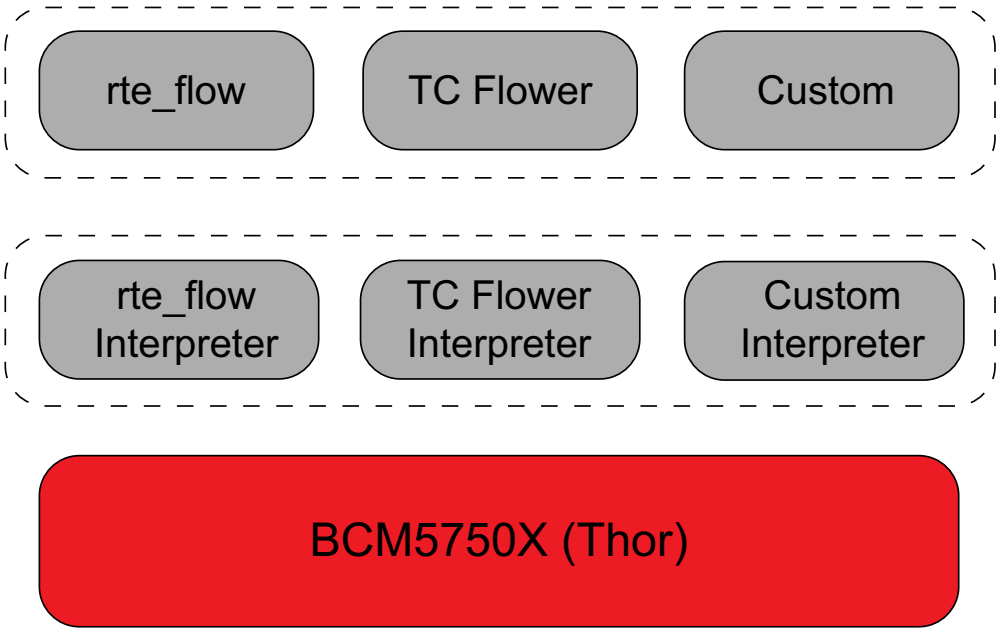
1 Overview

TruFlow™ is the hardware offload of a packet flow classifier on Broadcom BCM5750X (Thor) Ethernet adapters. This allows control of packets based on flows determined by matching well-known packet fields and metadata. TruFlow relies on the CFA (Configurable Flow Accelerator) hardware block in the ASIC which provides the TruFlow features.

The flow classification can be described either by OpenFlow implemented in the Open vSwitch package or by the TC Flower implemented in the iproute2 package. Offload of the flow classifier provides a mechanism to both increase throughput and reduce CPU utilization for users of flow-based systems.

This document provides instructions to enable and configure TruFlow. The instructions for enabling the TruFlow engine in the ASIC are provided in Linux through the TC Flower interface, DPDK's `rte_flow` API, or through a custom interface. In this document, TC Flower is used to demonstrate TruFlow with minimal overhead.

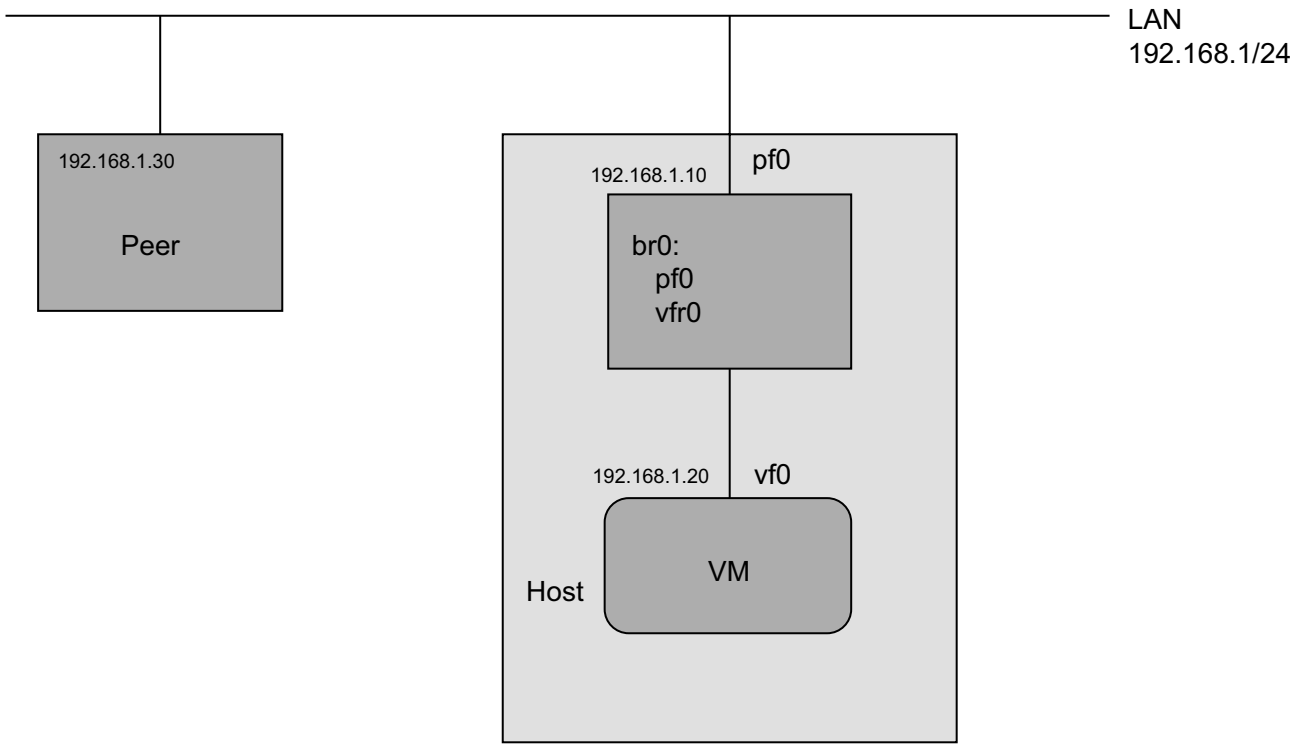
Figure 1: TruFlow Block Diagram



2 System Configuration

The following diagram provides a basic system configuration.

Figure 2: System Configuration



- The system configuration consists of two PCs: Peer and Host.
- The Host is equipped with a Broadcom BCM5750X (Thor) Ethernet network adapter with the latest OOB driver, which is configured to demonstrate TruFlow.
- Both the Peer and Host are connected to a common switch.
- A VM is created on the Host to demonstrate the use of TruFlow to control traffic between the Peer and the VM.
- To add networking capability to the VM, bridge br0 was created, as well as a virtual function VF0 on the physical function PF0 and a representor VFR0.
- PF0 and the representor VFR0 were added to the bridge br0 as well as VF0 to the VM.

All commands in this document must be run with superuser privileges.

3 Enabling TruFlow

To enable TruFlow, SR-IOV, and to set the global resource strategy to static in the network adapter firmware use the bnxtnvm utility. In this case, BCM5750X is installed in the PCIe slot 01:00.0 and the PF0 is enp1s0npf0.

NOTE: All highlighted portions must be updated according to the settings of the test system.

3.1 Enabling Truflow on the Host

To enable TruFlow on the host:

```
PF_iface=enp1s0npf0
bnxtnvm -dev=$PF_iface setoption=enable_truflow:0#0x01
# Enable truflow on port1 (second port). Cmd not needed for single-pf boards
bnxtnvm -dev=$PF_iface setoption=enable_truflow:1#0x01
bnxtnvm -dev=$PF_iface setoption=enable_sriov#1
bnxtnvm -dev=$PF_iface setoption=afm_rm_resc_strategy:#1
reboot
```

Expected behavior:

- Enumerated BCM5750X device is seen using lspci.
- All bnxtnvm commands are executed successfully.

4 Configuring the Network on the Host

The networking environment is configured as depicted in System Configuration. This section is divided into the following major steps:

- Setting Up the VF
- Setting Up the Representor
- Setting Up the Bridge

4.1 Setting Up the VF

To enable trusted virtual function VF0 on PF0 (enp1s0npf0) on the host:

4.1.1 Enabling the VF on the Host

To set up the VF on the host:

```
PF_iface=enp1s0npf0
ip link set $PF_iface up
echo 1 > /sys/class/net/$PF_iface/device/sriov_numvfs

# the following command is required only when used with DPDK

bnxtnvm -dev=$PF_iface vf 0 trust enable
```

Expected behavior:

- Link on the PF interface is up
- VF0 with “trust on” is set on PF0 (example: enp1s0npf0) and a new interface is set for VF0 (example: enp2s0v0).

NOTE: In this case, VF0 is enumerated as device 02:00.0.

4.1.2 Disabling the VF on the Host

To disable the VF on the host:

```
echo 0 > /sys/class/net/$PF_iface/device/sriov_numvfs
```

4.2 Setting Up the Representor

To add a representor VFR0, use the devlink utility. In this case, the created representor interface is eth0:

4.2.1 Adding a Representor on the Host

To add a representor on the host:

```
# PF's domain/bus/device/function ID
PF_bdf=0000:01:00.0
devlink dev eswitch set pci/$PF_bdf mode switchdev
```

Expected behavior:

- "`devlink dev eswitch show pci/$PF_bdf`" should show mode switchdev
- "`ip link`" should show a new VFR0 (eth0) interface

4.2.2 Removing a Representor from the Host

To remove a representor from the host:

```
devlink dev eswitch set pci/$PF_bdf mode legacy
devlink dev eswitch show pci/$PF_bdf
```

4.3 Setting Up the Bridge

Create a bridge br0 and add PF0 (enp1s0npf0) with representor VFR0(eth0):

4.3.1 Creating a Bridge on the Host

To create a bridge on the host:

```
PF_iface=enp1s0npf0
VFR_iface=eth0
BR_IP=192.168.1.10
ip link add br0 type bridge
ip addr flush dev $PF_iface
ip addr add $BR_IP/24 dev br0
ip link set $PF_iface master br0
ip link set $VFR_iface master br0
ip link set br0 up
```

Expected behavior:

- Bridge br0 is up and working.
- At this point, ping the bridge IP on the host (192.168.1.10) from the Peer and vice versa.

4.3.2 Removing a Bridge from the Host

To remove a bridge from the host:

```
ip link set dev br0 type bridge stp_state 0
ip link set $VFR_iface nomaster
ip link set $PF_iface nomaster
ip link del br0
```

5 Creating a VM

This section provides instructions to install Ubuntu VM inside CentOS8 Host. The following example relies on the virt-manager's supporting tools (https://virt-manager.org/). However, the same exercise can be accomplished using other virtual machine frameworks such as Oracle VirtualBox or VMware ESX.

5.1 Creating a VM on the Host

To create a VM on the host:

```
sed -i 's/SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config
reboot
dnf groupinstall "Virtualization Host"
dnf install virt-install
systemctl start libvirtd
mkdir -pv /kvm/{disk,iso}
cd /kvm/iso/
wget http://releases.ubuntu.com/22.04/ubuntu-22.04.1-desktop-amd64.iso

# Note the virt-install command spans multiple lines in this document
# It should be a single command on the terminal
virt-install --name udesktop22_04-01 --os-type linux --os-variant ubuntu22.04 --ram 4096 --disk /kvm/
disk/udesktop22_04-01.img,device=disk,bus=virtio,size=20,format=qcow2 --graphics vnc,listen=0.0.0.0
--noautoconsole --hvm --cdrom /kvm/iso/ubuntu-22.04.1-desktop-amd64.iso --boot cdrom,hd

virsh start udesktop22_04-01

virsh vncdisplay udesktop22_04-01

# At this point the user can launch VNC viewer from an external PC,
# and connect to the VNC server on the virtual machine that we created.

# Complete the OS setup through the VNC client
```

5.2 Removing a VM from the Host

To remove a VM from the host:

```
virsh destroy udesktop22_04-01
virsh undefine --remove-all-storage udesktop22_04-01
rm -rf /kvm/
```

NOTE: The standard VM installation comes with default NAT interfaces virbr0 and vnet0 on subnet 192.168.122/24, which is not used. Use subnet 192.168.1/24 for testing.

6 Configuring Networking on the VM

To configure networking on the VM, add VF0 to the VM. It is on PCIe slot 02:00.0, which was described in vf0.xml. This file needs must be created. Note the address domain portion of the XML file must match the domain/bus/device/function portion of VF0.

6.1 Creating the vf0.xml File

To create the vf0.xml file:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <driver name='vfio'/>
  <source>
    <address domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
  </source>
</hostdev>
```

6.2 Adding VF0 to the VM

Use the virsh utility on the Host to add VF0 to the VM from the created file vf0.xml.

```
virsh shutdown udesktop22_04-01
virsh attach-device udesktop22_04-01 vf0.xml --config
virsh start udesktop22_04-01
```

Sample log output on the VM terminal:

```
brcm@brcm-KVM:~$ lshw -businfo -class network

WARNING: you should run this program as super-user.

Bus info          Device      Class          Description
=======================================================
pci@0000:01:00.0              network        Virtio network device
virtio@0          enp1s0      network        Ethernet interface
pci@0000:07:00.0  enp7s0      network        BCM5750X NetXtreme-E Ethernet Virtual Function
```

Expected behavior:

- The VF attached to the VM should be enumerated as a separate interface. In the previous example, this is enp7s0 enumerated as 0000:07:00.0
- The virtio network interface (enp1s0 in the previous example) is not used. Use the interface that is associated with the VF.
- The IP addresses can be successfully assigned to this VF interface and can ping the Peer (and the Peer can ping the VF interface on the VM).

7 Testing TruFlow

In this section, TruFlow is tested using the TC flower utility.

VM Ingress:

- A filter is demonstrated that rewrites the source address of ping echo request packets.
- A filter is also demonstrated which drops all IP traffic to the VM.

VM Egress:

- A filter is demonstrated that rewrites source address of ping echo reply packets.
- A filter is also demonstrated which drops all IP traffic to the Peer.

To begin the test:

1. Start a ping from the Peer to the VM.
2. Run `tcpdump` on the VM.
3. Continue with the following sections.

7.1 Host (VM Ingress)

This test (VM ingress) is executed as follows:

```
PF_iface=enp1s0npf0
VFR_iface=eth0
VM_IP=192.168.1.20
PEER_IP=192.168.1.30

# add device to the queue
tc qdisc add dev $PF_iface ingress

# add a filter to rewrite source IP address
tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action pedit ex munge
ip src set 33.33.33.33 pipe action mirred egress redirect dev $VFR_iface

# show the filter
tc filter show dev $PF_iface ingress

# remove the filter
tc filter del dev $PF_iface ingress

# add a filter to drop traffic
tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action drop

# show the filter
tc filter show dev $PF_iface ingress

# remove the filter
tc filter del dev $PF_iface ingress

# remove device from the queue
tc qdisc del dev $PF_iface ingress
```

7.2 Host (VM Egress)

The test (VM egress) is executed as follows:

```
PF_iface=enp1s0npf0
VFR_iface=eth0
VM_IP=192.168.1.20
PEER_IP=192.168.1.30

# add device to the queue
tc qdisc add dev $VFR_iface ingress

# add a filter to rewrite source IP address
tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action pedit ex
munge ip src set 44.44.44.44 pipe action mirred egress redirect dev $PF_iface

# show the filter
tc filter show dev $VFR_iface ingress

# remove the filter
tc filter del dev $VFR_iface ingress

# add a filter to drop traffic
tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action drop

# show the filter
tc filter show dev $VFR_iface ingress

# remove the filter
tc filter del dev $VFR_iface ingress

# remove device from the queue
tc qdisc del dev $VFR_iface ingress
```

NOTE: The `skip_sw` argument ensures that the flow is executed in hardware.

Expected behavior:

- The `in_hw` field in the output of `tc filter show dev $PF_iface` ingress indicates that the flow is in the hardware.
- When the filter is added, the tcpdump and the ping outputs differ based on the applied filter.
- When the source IP is changed with the ingress filter, the source IP changes for ICMP echo requests in the tcpdump on the VM.
- When the source IP is changed with the egress filter, the source IP changes for ICMP echo reply in ping output on the Peer.
- When traffic is dropped, the ping output stops and tcpdump stops showing ICMP packets.

8 Sample Logs

This section provides sample logs from the test.

8.1 Host Log – Network Configuration

The host log of the setup is as follows:

```
# ifconfig enp1s0npf0 up
# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff

# echo 1 > /sys/class/net/enp1s0npf0/device/sriov_numvfs
# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
    vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto,
trust off
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff

# ./bnxtnvm -dev=enp1s0npf0 vf 0 trust enable

Command Executed Successfully.

# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
    vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto,
trust on
```

```
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff

# devlink dev show

pci/0000:01:00.0
pci/0000:02:00.0

# devlink dev eswitch set pci/0000:01:00.0 mode switchdev
# devlink dev eswitch show pci/0000:01:00.0

pci/0000:01:00.0: mode switchdev

# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
    vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto,
trust on
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff
6: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN mode DEFAULT group
default qlen 1000
    link/ether 00:62:0b:6d:12:5e brd ff:ff:ff:ff:ff:ff

# ip link add br0 type bridge
# ip addr add 192.168.1.10/24 dev br0
# ip link set eth0 master br0
# ip link set enp1s0npf0 master br0
# ip link set br0 up
# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br0 state UP mode DEFAULT
group default qlen 1000
    link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
    vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto,
trust on
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff
6: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UNKNOWN mode
DEFAULT group default qlen 1000
    link/ether 00:62:0b:6d:12:5e brd ff:ff:ff:ff:ff:ff
```

```
7: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
qlen 1000
    link/ether 00:62:0b:6d:12:5e brd ff:ff:ff:ff:ff:ff
```

8.2 Host Log – Testing TruFlow

The host log of the test is as follows:

```
# PF_iface=enp1s0f0np0
# VFR_iface=eth0
# VM_IP=192.168.1.20
# PEER_IP=192.168.1.30
# tc qdisc add dev $PF_iface ingress
# tc qdisc show dev $PF_iface ingress
qdisc ingress ffff: parent ffff:fff1 ----------------
# tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action pedit ex
munge ip src set 33.33.33.33 pipe action mirred egress redirect dev $VFR_iface
# tc filter show dev $PF_iface ingress
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.20
  skip_sw
  in_hw in_hw_count 1
    action order 1:  pedit action pipe keys 1
     index 1 ref 1 bind 1
     key #0  at ipv4+12: val 21212121 mask 00000000
    used_hw_stats delayed

    action order 2: mirred (Egress Redirect to device eth0) stolen
    index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $PF_iface ingress
# tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action drop
# tc filter show dev $PF_iface ingress
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.20
  skip_sw
  in_hw in_hw_count 1
    action order 1: gact action drop
     random type none pass val 0
     index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $PF_iface ingress
# tc qdisc del dev $PF_iface ingress
# tc qdisc add dev $VFR_iface ingress
# tc qdisc show dev $VFR_iface ingress
qdisc ingress ffff: parent ffff:fff1 ----------------
# tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action pedit
ex munge ip src set 44.44.44.44 pipe action mirred egress redirect dev $PF_iface
# tc filter show dev $VFR_iface ingress
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.30
```

```
  skip_sw
  in_hw in_hw_count 1
    action order 1:  pedit action pipe keys 1
     index 1 ref 1 bind 1
     key #0  at ipv4+12: val 2c2c2c2c mask 00000000
    used_hw_stats delayed

    action order 2: mirred (Egress Redirect to device enp9s0f0np0) stolen
    index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $VFR_iface ingress
# tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action drop
# tc filter show dev $VFR_iface ingress
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.30
  skip_sw
  in_hw in_hw_count 1
    action order 1: gact action drop
     random type none pass val 0
     index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $VFR_iface ingress
# tc qdisc del dev $VFR_iface ingress
```

8.3 VM Log

The VM log is as follows:

```
brcm@brcm-KVM-22:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:8b:c1:47 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.236/24 brd 192.168.122.255 scope global dynamic noprefixroute enp1s0
       valid_lft 2306sec preferred_lft 2306sec
    inet6 fe80::1caf:a751:41e:9642/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 3a:0a:cb:72:1b:a6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.20/24 brd 192.168.1.255 scope global enp7s0
       valid_lft forever preferred_lft forever
brcm@brcm-KVM-22:~$ sudo tcpdump -ni enp7s0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:13:25.919610 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:13:25.919622 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:13:26.377695 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 8, length 64
14:13:26.377867 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 8, length 64
14:13:27.402244 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 9, length 64
```

```
14:13:27.402418 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 9, length 64

... (above is a nornal traffic showing ICMP and STP packets)

14:13:33.962551 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:13:33.962561 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:13:34.473590 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 16, length 64
14:13:35.497581 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 17, length 64
14:13:35.970049 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:13:35.970070 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:13:36.521562 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 18, length 64
14:13:37.546220 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 19, length 64

... (above is a traffic after with ingress filter rewriting the ping request source address)

14:13:59.050013 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 40, length 64
14:13:59.050178 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 40, length 64
14:14:00.073636 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 41, length 64
14:14:00.073794 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 41, length 64
14:14:00.109307 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:00.109324 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36

... (above is a normal traffic after the filter was removed)

14:14:18.346006 ARP, Request who-has 192.168.1.20 tell 192.168.1.30, length 46
14:14:18.346096 ARP, Reply 192.168.1.20 is-at 3a:0a:cb:72:1b:a6, length 28
14:14:20.219290 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:20.219309 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:22.228215 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:22.228236 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:24.247132 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:24.247155 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:26.265764 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:26.265777 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36

... (above is a traffic when all IP packets to VM were dropped)

14:14:44.969832 ARP, Request who-has 192.168.1.20 tell 192.168.1.30, length 46
14:14:44.969841 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 85, length 64
14:14:44.969934 ARP, Reply 192.168.1.20 is-at 3a:0a:cb:72:1b:a6, length 28
14:14:44.970105 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 85, length 64
14:14:45.993826 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 86, length 64
14:14:45.993984 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 86, length 64
```

```
14:14:46.380958 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36
14:14:46.380978 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id
8001.8c:47:be:df:82:80.8100, length 36

...(above is a normal traffic after the filter was removed)
...
^C
517 packets captured
517 packets received by filter
0 packets dropped by kernel
brcm@brcm-KVM-22:~$
```

8.4 Peer Log

The peer log is as follows:

```
brcm@peer:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether d0:94:66:42:61:59 brd ff:ff:ff:ff:ff:ff
    altname enp1s0f0
    inet 10.136.14.59/24 brd 10.136.14.255 scope global eno1
       valid_lft forever preferred_lft forever
    inet6 fe80::d294:66ff:fe42:6159/64 scope link
       valid_lft forever preferred_lft forever
4: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether d0:94:66:42:61:5a brd ff:ff:ff:ff:ff:ff
    altname enp1s0f1
39: enp2s0np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 5000 qdisc mq state UP group default qlen 1000
    link/ether 00:0a:f7:31:43:f0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.30/24 brd 192.168.1.255 scope global enp2s0np0
       valid_lft forever preferred_lft forever
    inet6 fe80::20a:f7ff:fe31:43f0/64 scope link
       valid_lft forever preferred_lft forever
brcm@peer:~$ ping 192.168.1.20

PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=1 ttl=64 time=0.636 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=64 time=0.393 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=64 time=1.08 ms
64 bytes from 192.168.1.20: icmp_seq=4 ttl=64 time=0.397 ms
64 bytes from 192.168.1.20: icmp_seq=5 ttl=64 time=0.397 ms
64 bytes from 192.168.1.20: icmp_seq=6 ttl=64 time=0.431 ms
64 bytes from 192.168.1.20: icmp_seq=7 ttl=64 time=0.656 ms
64 bytes from 192.168.1.20: icmp_seq=8 ttl=64 time=0.540 ms
64 bytes from 192.168.1.20: icmp_seq=9 ttl=64 time=1.12 ms
64 bytes from 192.168.1.20: icmp_seq=10 ttl=64 time=0.960 ms
64 bytes from 192.168.1.20: icmp_seq=11 ttl=64 time=0.685 ms
64 bytes from 192.168.1.20: icmp_seq=12 ttl=64 time=0.463 ms
64 bytes from 192.168.1.20: icmp_seq=13 ttl=64 time=1.06 ms
64 bytes from 192.168.1.20: icmp_seq=14 ttl=64 time=0.463 ms
```

```
64 bytes from 192.168.1.20: icmp_seq=15 ttl=64 time=0.662 ms

(there is a gap in the initial icmp_seq when packets are dropped with ingress filter)

64 bytes from 192.168.1.20: icmp_seq=40 ttl=64 time=0.850 ms
64 bytes from 192.168.1.20: icmp_seq=41 ttl=64 time=0.474 ms
64 bytes from 192.168.1.20: icmp_seq=42 ttl=64 time=1.12 ms
64 bytes from 192.168.1.20: icmp_seq=43 ttl=64 time=0.990 ms
64 bytes from 192.168.1.20: icmp_seq=44 ttl=64 time=0.459 ms
64 bytes from 192.168.1.20: icmp_seq=45 ttl=64 time=1.12 ms
64 bytes from 192.168.1.20: icmp_seq=46 ttl=64 time=0.902 ms

... (normal ping continues)

64 bytes from 192.168.1.20: icmp_seq=126 ttl=64 time=1.17 ms
64 bytes from 192.168.1.20: icmp_seq=127 ttl=64 time=0.861 ms
64 bytes from 192.168.1.20: icmp_seq=128 ttl=64 time=0.466 ms
64 bytes from 44.44.44.44: icmp_seq=129 ttl=64 time=0.453 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=130 ttl=64 time=0.513 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=131 ttl=64 time=0.657 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=132 ttl=64 time=0.522 ms (DIFFERENT ADDRESS!)

... (the source address of ping reply packets was changed with egress filter)

64 bytes from 44.44.44.44: icmp_seq=150 ttl=64 time=0.399 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=151 ttl=64 time=0.463 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=152 ttl=64 time=0.447 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=153 ttl=64 time=0.650 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=154 ttl=64 time=0.458 ms (DIFFERENT ADDRESS!)
64 bytes from 192.168.1.20: icmp_seq=155 ttl=64 time=1.16 ms
64 bytes from 192.168.1.20: icmp_seq=156 ttl=64 time=1.02 ms
64 bytes from 192.168.1.20: icmp_seq=157 ttl=64 time=0.961 ms
64 bytes from 192.168.1.20: icmp_seq=158 ttl=64 time=0.443 ms

... (ping replies are back to normal after egress filter was removed)
...
^C
--- 192.168.1.20 ping statistics ---
197 packets transmitted, 142 received, 27.9188% packet loss, time 199643ms
rtt min/avg/max/mdev = 0.388/0.684/1.356/0.272 ms
brcm@peer:~$
```

9 Supported Patterns and Actions

The following patterns and actions are supported by TruFlow:

Table 1: Patterns and Actions

Match Patterns [* means OPTIONAL]	Name		
[VLAN/VNI]*, DMAC/MASK, SMAC/MASK (Only IP packets, not pure L2 like ARP, ICMP, and so forth)	Inner L2		
[VLAN/VNI]*, DIP/MASK, SIP/MASK, PROTO/MASK, DPORT/MASK, SPORT/MASK	Inner IPv4		
[VLAN/VNI]*, DIP, SIP, PROTO, DPORT, SPORT	Inner IPv6		
Action Patterns	**Match Patterns**		
Ingress	**Inner L2**	**Inner IPv4**	**Inner IPv6**
[count],	Yes	Yes	Yes
[count, drop],	Yes	Yes	Yes
[count, pop_vlan],	Yes	Yes	Yes
[count, vxlan_decap],	Yes	Yes	Yes
Action Patterns	**Match Patterns**		
Ingress NAT	**Inner L2**	**Inner IPv4**	**Inner IPv6**
[count, set_ipv4_src],	N/A	Yes	No
[count, set_ipv4_src, set_tp_src],	N/A	Yes	No
[count, set_ipv4_dst],	N/A	Yes	No
[count, set_ipv4_dst, set_tp_dst],	N/A	Yes	No
Action Patterns	**Match Patterns**		
Egress	**Inner L2**	**Inner IPv4**	**Inner IPv6**
[drop, count],	Yes	Yes	Yes
[count],	Yes	Yes	Yes
[count, set_vlan_pcp, set_vlan_vid, push_vlan],	Yes	Yes	Yes
[count, set_vlan_vid, push_vlan],	Yes	Yes	Yes
[vxlan_encap, count]	Yes	Yes	Yes
Action Patterns	**Match Patterns**		
Egress NAT	**Inner L2**	**Inner IPv4**	**Inner IPv6**
[count, set_ipv4_src],	N/A	Yes	No
[count, set_ipv4_src, set_tp_src],	N/A	Yes	No
[count, set_ipv4_dst],	N/A	Yes	No
[count, set_ipv4_dst, set_tp_dst],	N/A	Yes	No

10 Conclusion

Instructions for enabling TruFlow using TC Flower from iproute2 have been shown. It is also possible to describe TruFlow using OpenFlow from the Open vSwitch package. The following example shows the differences between commands using TC Flower and OpenFlow:

```
# Exact Match

TC: tc filter add dev <vfr0> protocol ip parent ffff: prio 10 chain 0 flower skip_sw src_mac
00:01:22:33:81:60 dst_mac 00:0a:f7:a6:81:60 action mirred egress redirect dev <pf0>

OF: ovs-ofctl add-flow ovsbr0 ""in_port=vfrep1 dl_dst=00:0a:f7:98:38:60 dl_src=c2:a1:ed:fe:a5:90
dl_type=0x0800 ipv4 actions=output:trustedvf""

# Wild Card

TC: tc filter add dev <vfr0> protocol ip parent ffff: prio 10 chain 0 flower skip_sw src_mac
00:01:22:33:81:60/ff:ff:ff:ff:ff:00 dst_mac 00:0a:f7:a6:81:60/ff:ff:ff:ff:ff:00 action mirred egress
redirect dev <pf0>

OF: ovs-ofctl add-flow ovsbr0 ""in_port=vfrep1 dl_dst=00:0a:f7:98:38:60/ff:ff:ff:ff:ff:00
dl_src=c2:a1:ed:fe:a5:90/ff:ff:ff:ff:ff:00 dl_type=0x0800 ipv4 actions=output:trustedvf"""
```

11 References

This document refers to the following references:

- https://man7.org/linux/man-pages/man8/tc-flower.8.html
- https://www.openvswitch.org
- https://www.linux.com/training-tutorials/tc-show-manipulate-traffic-control-settings/
- https://www.linux.com/training-tutorials/qos-linux-tc-and-filters/
- https://linuxhint.com/install_kvm_qemu_centos_8/
- https://ubuntu.com/server/docs/virtualization-libvirt
- https://virt-manager.org/

Revision History

5750X-AN200; February 2, 2023

Initial release.