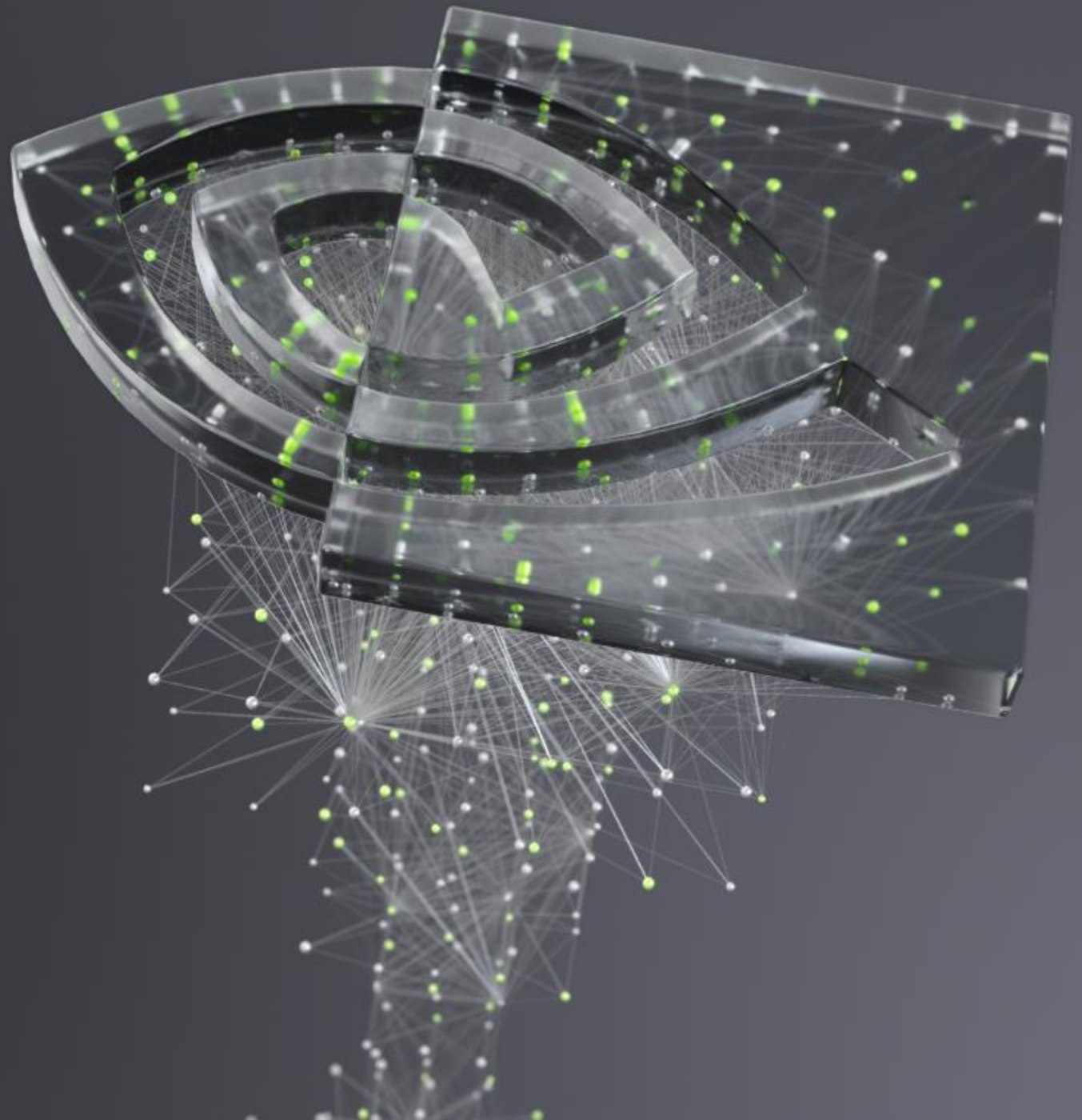


TC CONNECTION TRACKING

Oz Shlomo – ozsh@nvidia.com

Paul Blakey – paulb@nvidia.com

COMMUNITY CREDITS

Connection tracking offload is available since Kernel 5.7

- Bi-weekly synch meetings lead by Aaron Conole from RH
  - Participations from RH, Mellanox, Netronome, Intel and Broadcom
  - Guest appearances by David Miller
- Marcelo Ricardo Leitner supporting the development and review
- Jiri Pirko for TC integration
- Pablo Neira Ayuso for nf flow table hardware offload

AGENDA

Connection tracking overview

Offloading established connections

Connection tracking hardware model

CONNECTION TRACKING

- Building block for stateful packet filtering
  - Finds the connection in DB or creates a new entry
  - Associates a CT state – new, established, related
  - Initializes user defined mark and label value
  - Performs NAT – flagged with snat/dnat ct state
  - Validates the packets – controlled with tcp_liberal flag
- Used by multiple user-space applications
  - iptables, nft and openvswitch

CONNECTION TRACKING IN TC

Classifier and action

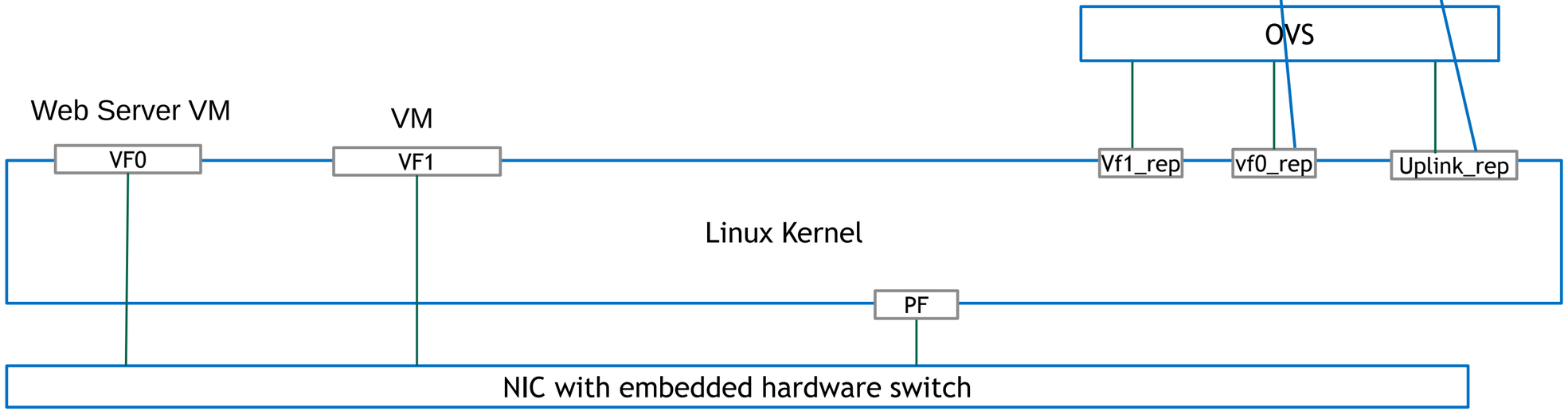
- TC action ct looks up the connection's CT state
  - Uses netfilter conntrack module
  - Performs NAT
  - Sets the ct_state, mark, label
- Flower classifier matching
  - Fields: zone, ct_state, mark, label
- Introduced in kernel 5.3
  - net/sched: Introduce tc connection tracking series

CONNECTION TRACKING USE CASE USING TC

Allow ingress traffic on tcp dst port 80

```
tc filter add dev uplink_rep ingress prio 1 chain 1 proto ip flower ct_state -trk ip_proto tcp dst_port 80 action ct pipe action goto chain 2
tc filter add dev uplink_rep ingress chain 1 prio 1 proto ip flower ct_state +new+trk action ct commit pipe action mirred egress redirect dev vf0_rep
tc filter add dev uplink_rep ingress chain 1 prio 1 proto ip flower ct_state -new+est+trk action mirred egress redirect dev vf0_rep
```

```
tc filter add dev vf0_rep ingress prio 1 chain 1 proto ip flower ct_state -trk ip_proto tcp src_port 80 action ct pipe action goto chain 2
tc filter add dev vf0_rep ingress chain 1 prio 1 proto ip flower ct_state +est+trk action mirred egress redirect dev uplink_rep
tc filter add dev vf0_rep ingress chain 1 prio 1 proto ip flower ct_state +new+trk action drop
```



CONNECTION TRACKING OFFLOAD

Offloading only established connections

- Connection setup and teardown is handled by software
- Connection aging is managed by the software
- Established connections are offloaded

OFFLOADING ESTABLISHED CONNECTIONS

Design alternatives

- Possible entry points for offloading established events
  - From nf conntrack
  - From act_ct
  - Using nf flowtable – available since kernel 4.16
- Integrates with nf conntrack
  - nf flowtable owns the connection and manages its aging
- Can be accelerated in hardware
  - Common platform to serve both nft and tc hardware offload

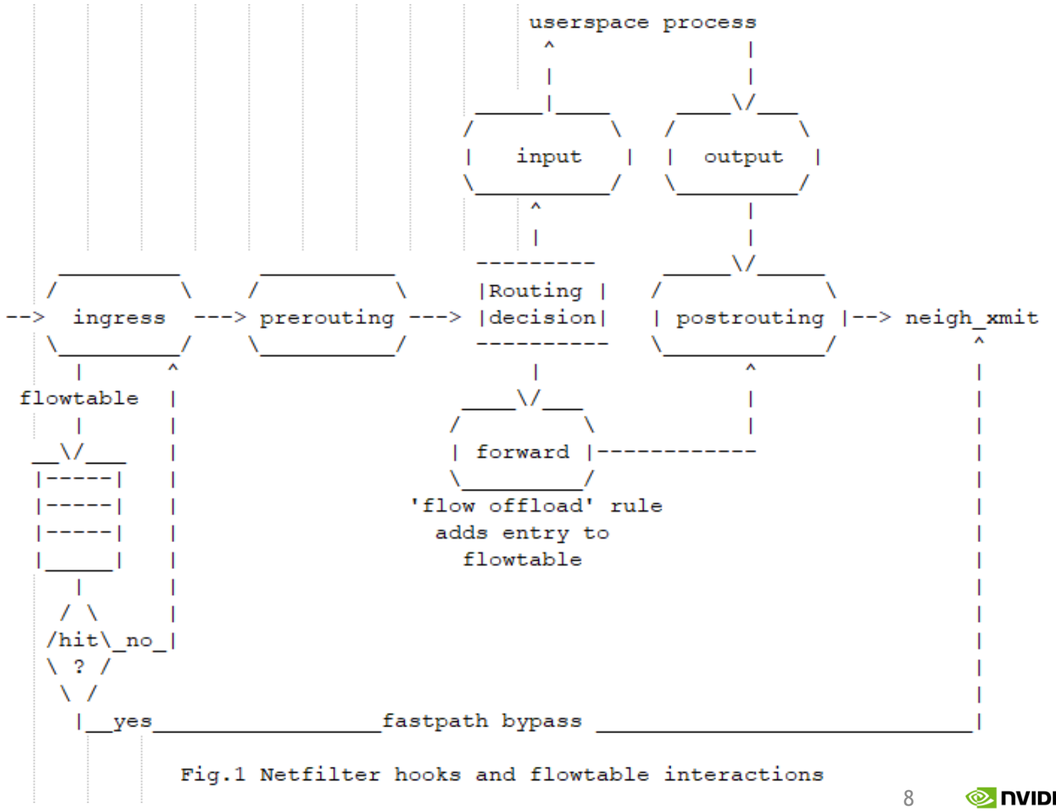


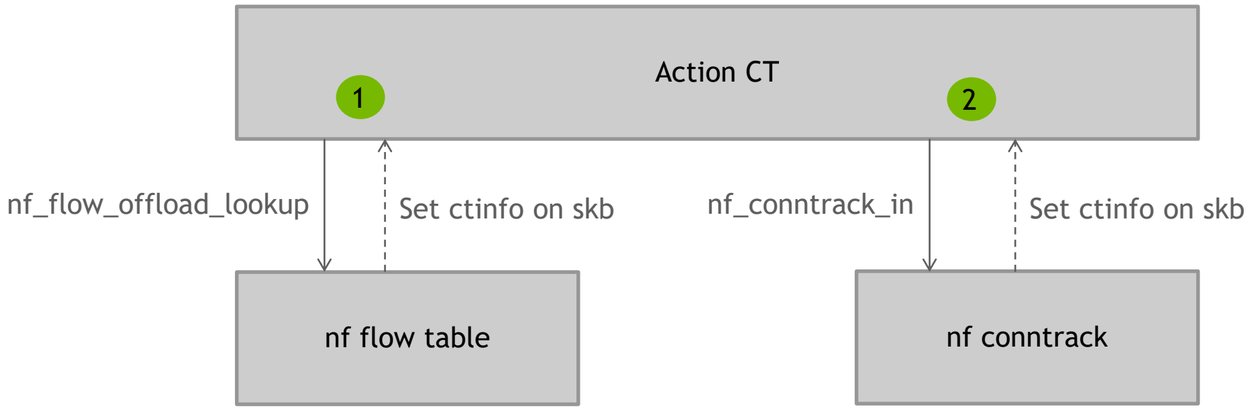
Fig.1 Netfilter hooks and flowtable interactions

NVIDIA

OFFLOADING ESTABLISHED CONNECTIONS

netfilter flow tables

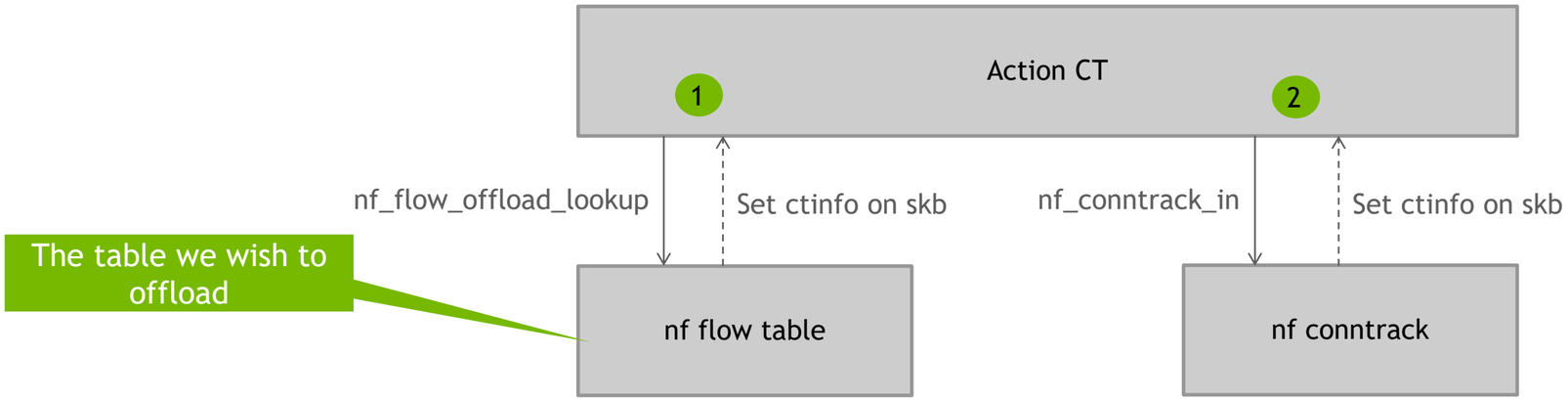
- Manage the established connections in a netfilter flow table
  - Lookup a <ct_zone, ip proto, src ip, dst ip, src port, dst port>6-tuple
  - On hit – set the ct_state, mark, label and performs NAT
  - Delete aged connections



OFFLOADING ESTABLISHED CONNECTIONS

netfilter flow tables

- Manage the established connections in a netfilter flow table
  - Lookup a <ct_zone, ip proto, src ip, dst ip, src port, dst port>6-tuple
  - On hit – set the ct_state, mark, label and performs NAT
  - Delete aged connections



MANAGING FLOW TABLES

Act CT implementation

- Flow tables are created per zone
  - Shared (referenced counted) by all CT action instances
- Flow tables are deleted on the last action reference
  - act_ct module reference is incremented per flow table
- Flows are added when connections are in established state
  - 5-tuple for both directions, NAT indication, nf_conn instance
  - Nf flow tables owns the flow
- Aging is managed by nf flow tables
- Introduced in kernel 5.6
  - act_ct: Software offload of conntrack_in

OFFLOADING ESTABLISHED CONNECTIONS

Hardware offload input parameters

- Flow offload instance is added to the flow table
  - Match parameters: zone, 5-tuple
  - Action parameters: CT metadata parameters (ct_state, mark, label) , NAT header rewrites
- Tuples that are offloaded by act_ct perform the following actions
  - (new) meta action – mark, label, reference to nf_ct_conn object
  - NAT mangle action (src/dst ip, src/dst port)

OFFLOADING ESTABLISHED CONNECTIONS

Driver notification

- Nf flow table callbacks registration is managed by the driver
  - Act CT is not aware of the net devices
  - Nf flow table is not aware of the devices
  - Tc is not aware of nf flow table
- Nf flow table exports nf_flow_table_offload_add_cb/nf_flow_table_offload_del_cb methods
  - FT instance is passed as act_ct flow offload parameter while offloading the filter
  - Managed using flow_block_cb object
- Driver manages its registration by ref counting offloaded act_ct actions
- Mellanox driver implementation was introduced in kernel 5.7
  - Introduce connection tracking series

OFFLOADING ESTABLISHED CONNECTIONS

Summary

- Current tc filter offload infrastructure does not provide a mechanism for offloading established flows
  - Act CT is partially processed by hardware (only established flows)
- Established connections are added to nf flow table instances
  - Software offload – bypassing nf conntrack
  - Hardware offload – notifying drivers of established connections
- Connection setup and teardown is managed by software
- Aging is managed by nf flow table
  - Hardware stats are retrieved using stats cb method

MONITORING OFFLOADED CONNECTIONS

- nf_conntrack procfs output
  - OFFLOAD – Connection is owned by nf flow table (software offload)
  - HW_OFFOAD – Connection is in hardware
- sudo cat /proc/net/nf_conntrack

```
ipv4     2 tcp      6 src=5.5.5.5 dst=5.5.5.9 sport=41460 dport=5201 src=5.5.5.9 dst=5.5.5.5 sport=5201 dport=41460 [OFFLOAD/HW_OFFLOAD] mark=0 zone=0 use=3

ipv4     2 tcp      6 src=5.5.5.5 dst=5.5.5.9 sport=41462 dport=5201 src=5.5.5.9 dst=5.5.5.5 sport=5201 dport=41462 [OFFLOAD/ HW_OFFLOAD] mark=0 zone=0 use=3
```

- HW_OFFLOAD flag was added to kernel 5.7
  - netfilter: nf_conntrack, add IPS_HW_OFFLOAD status bit

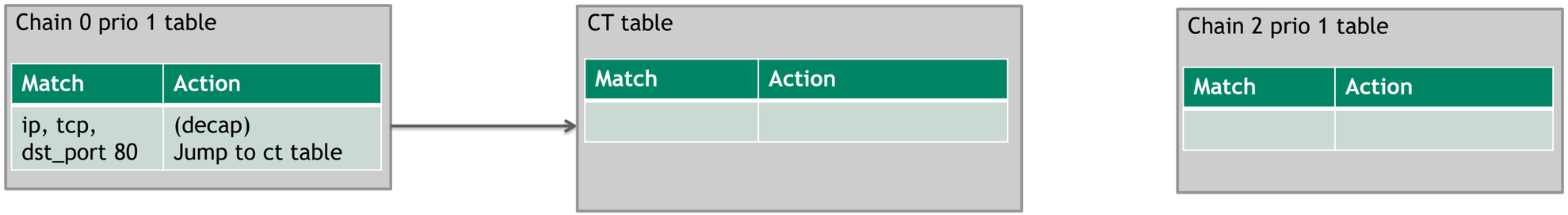
CONNECTION TRACKING HARDWARE MODEL

tc filter add dev uplink_rep ingress chain 0 prio 1 proto ip flower ct_state -trk ip_proto tcp dst_port 80 action ct pipe action goto chain 2

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state +new+trk action ct commit pipe action mirred egress redirect dev vf0_rep

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state -new+est+trk action mirred egress redirect dev vf0_rep

- Hardware offload follows the software model
  - Tc <chain, prio> tuple is mapped to a hw table
  - Nf flow table is mapped to a hardware table



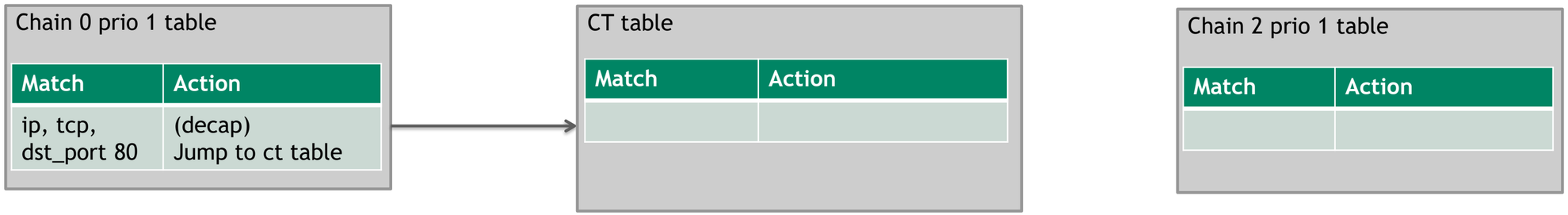
CONNECTION TRACKING HARDWARE MODEL

tc filter add dev uplink_rep ingress chain 0 prio 1 proto ip flower ct_state -trk ip_proto tcp dst_port 80 action ct pipe action goto chain 2

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state +new+trk action ct commit pipe action mirred egress redirect dev vf0_rep

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state -new+est+trk action mirred egress redirect dev vf0_rep

- Hardware offload follows the software model
  - Tc <chain, prio> tuple is mapped to a hw table
  - Nf flow table is mapped to a hardware table



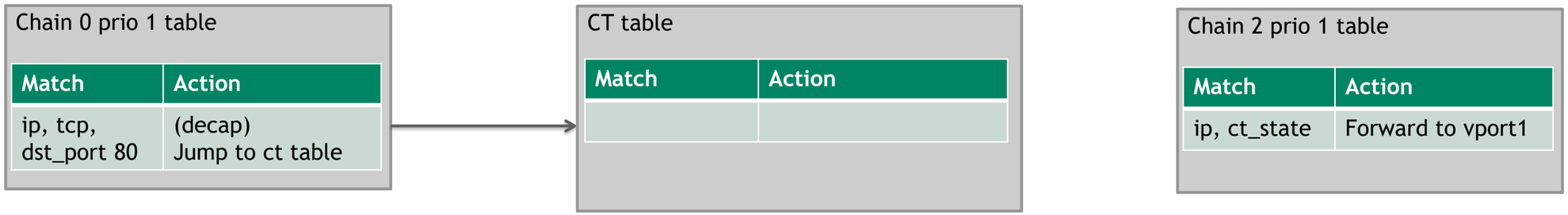
CONNECTION TRACKING HARDWARE MODEL

tc filter add dev uplink_rep ingress chain 0 prio 1 proto ip flower ct_state -trk ip_proto tcp dst_port 80 action ct pipe action goto chain 2

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state +new+trk action ct commit pipe action mirred egress redirect dev vf0_rep

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state -new+est+trk action mirred egress redirect dev vf0_rep

- Hardware offload follows the software model
  - Tc <chain, prio> tuple is mapped to a hw table
  - Nf flow table is mapped to a hardware table



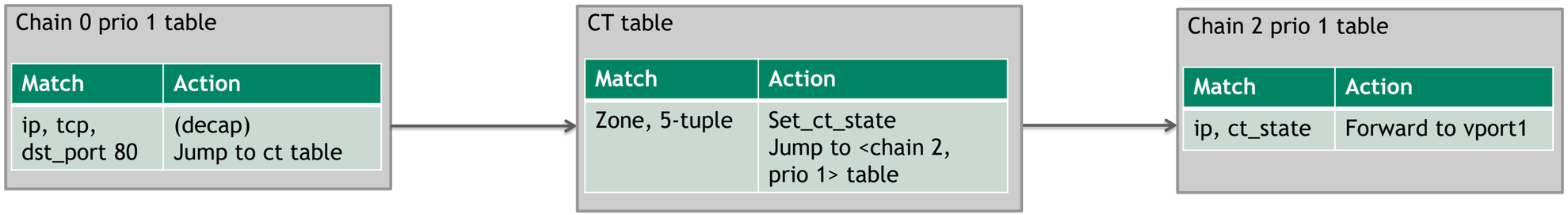
CONNECTION TRACKING HARDWARE MODEL

tc filter add dev uplink_rep ingress chain 0 prio 1 proto ip flower ct_state -trk ip_proto tcp dst_port 80 action ct pipe action goto chain 2

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state +new+trk action ct commit pipe action mirred egress redirect dev vf0_rep

tc filter add dev uplink_rep ingress chain 2 prio 1 proto ip flower ct_state -new+est+trk action mirred egress redirect dev vf0_rep

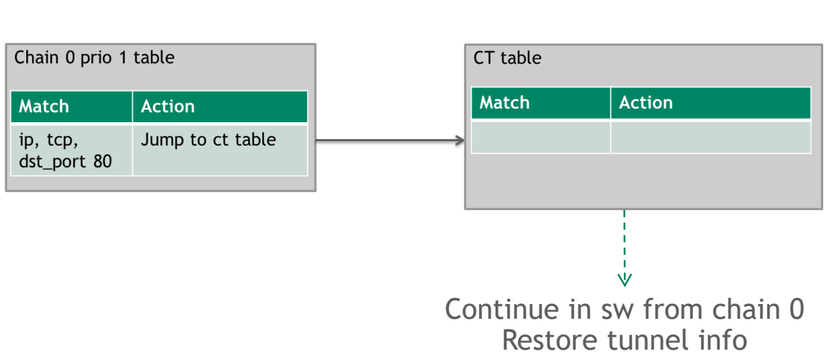
- Hardware offload follows the software model
  - Tc <chain, prio> tuple is mapped to a hw table
  - Nf flow table is mapped to a hardware table – established flow callback

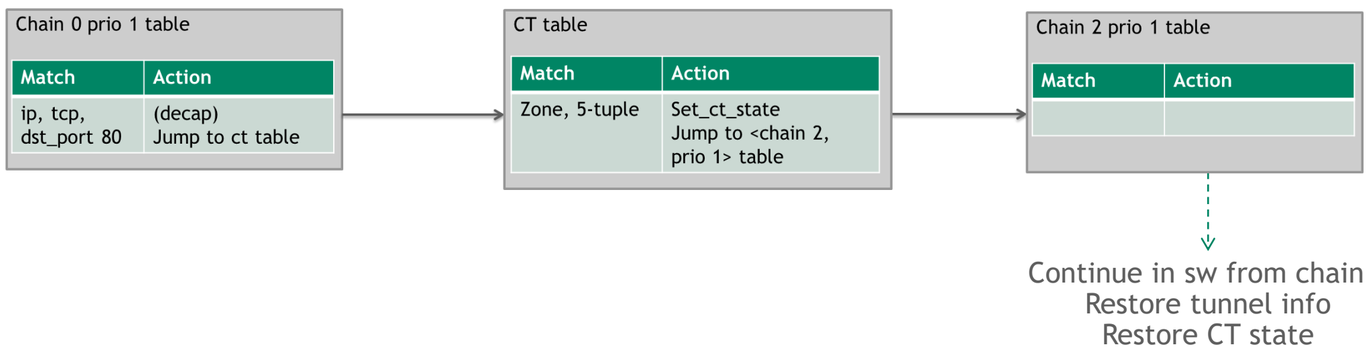


CONNECTION TRACKING HARDWARE MODEL

Hardware misses

- Software should continue where the hardware left off
  - Stats and packet mangling was executed by hardware
- Driver sets the chain that missed on (new) tc skb extension
- Driver restores the skb->tun_info structure





CONNECTION TRACKING OFFLOAD

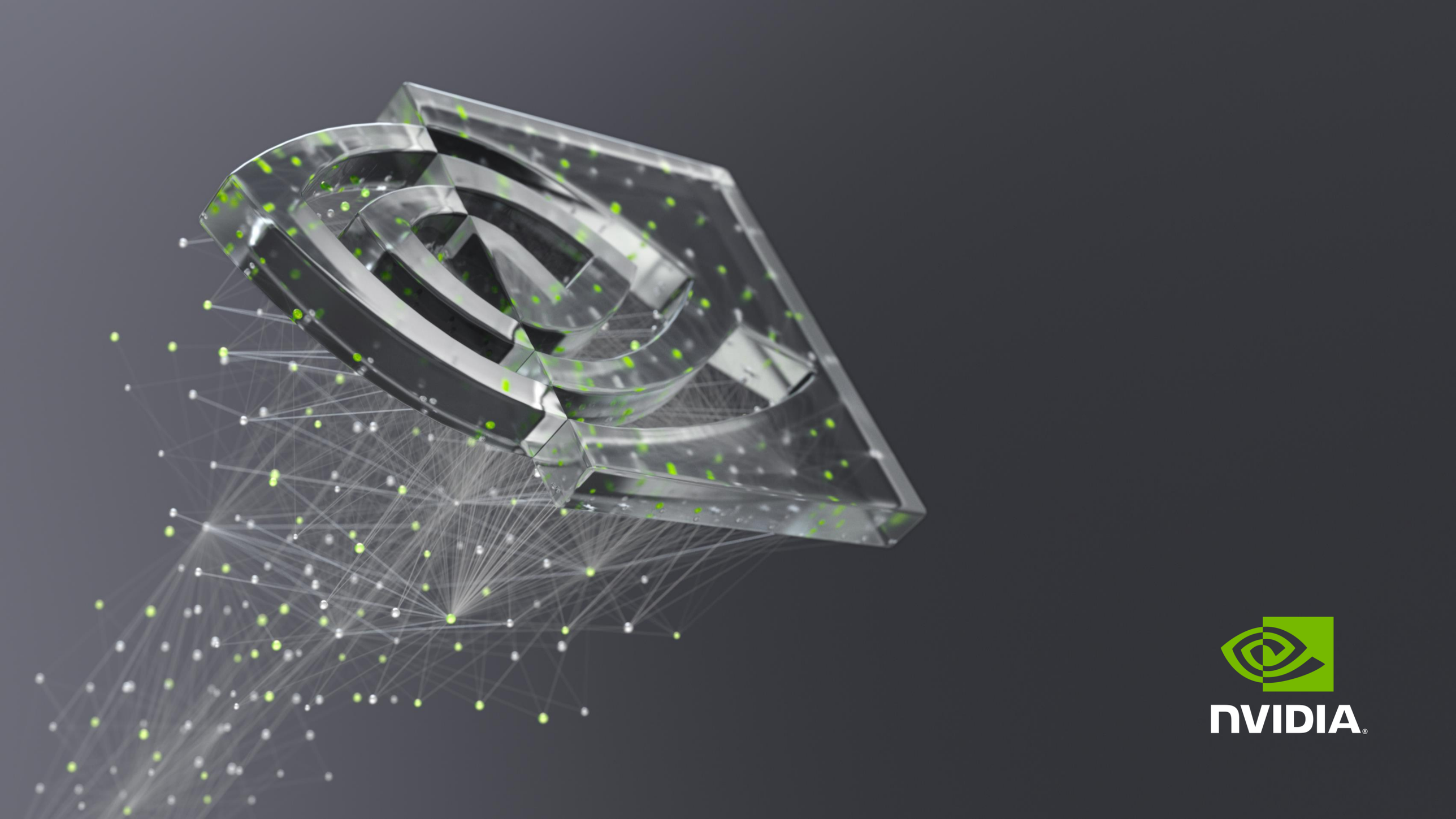
Openvswitch integration – handling tc misses

- tc is processed before the openvswitch rx handler
- Openvswitch should continue where tc left off
  - OVS recirc id is directly mapped to tc chain
- The tc skb extension for hw miss is reused to convey the tc chain that missed
  - Added in kernel 5.3 - net: openvswitch: Set OvS recirc_id from tc chain index

SUMMARY

Submitted patches

- net/sched: Introduce tc connection tracking
- net: openvswitch: Set OvS recirc_id from tc chain index
- netfilter flowtable hardware offload
- act_ct: Software offload of conntrack_in
- netfilter: nf_conntrack, add IPS_HW_OFFLOAD status bit
- Introduce connection tracking offload – Mellanox driver implementation


NVIDIA