mlxsw – TC offload

- Mellanox Spectrum ASIC
- cls_flower offload to TCAM supported upstream
  - Basic cls_flower keys
  - Drop, mirred-redirect, vlan-modify actions
- Non-optimal TCAM utilization
  - Lack of multi-table in TC (work in progress)
  - Need to hint driver about future rule format somehow
    - The key size defines number of TCAM rules

Packet sampling (Yotam Gigi)

psample netlink channel

- Before, it was possible to sample packets using NFLOG
- NFLOG is a netlink channel, which is used by netfilter
  - It transfers packets to userspace
- A new netlink (generic) channel, dedicated for packet sampling
  - Not bound to netfilter
  - Can be done used by tc action
- Similarly to NFLOG, there can be several packet sampling done in the system, each to a different psample_group
- Packets are sent to userpspace via the “sample” generic Netlink multicast group
- Each packet contains the psamle_group id

sample tc action

- Peeks packets randomly and send them through the psample netlink channel
- Example of usage:

  tc filter add dev eth0 parent ffff: matchall \
  action sample rate 12 group 13
- This rule is offloaded to HW if eth0 is mlxsw driver instance

TC multitable

Current state

- Each qdisc maintains one chain of filters
  - `struct tcf_proto __rcu *filter_list` in private struct
    - This is in `struct net_device` in case of ingress/clsact
  - `Qdisc_class_ops->tcf_chain` op to get it from within the qdisc private by cls_api.c
- To process the chain, qdisc calls `tc_classify`
  - `tc_classify` walks the chain
    - calls struct tcf_proto->classify for each node
  - `classify` may return TC_ACT_RECLASSIFY which causes the chain to be walked again (there is a limit of number of reclassifications)
- One chain ~ one “table”
- One qdisc ~ one chain

Multi-table motivation

- To be able to assign a pipeline instead of single flat table
  - For offload, helps to utilize HW better
- Multiple qdisc with single chain vs. one qdisc with multiple chains
  - Single qdisc variant looks nicer
- Side effect of changes would be ability to share chains between multiple qdiscs

Plan

- Introduce a “block” to hold many chains
  - Holds `list_head` of chains, each chain has `u32` index
  - `struct tcf_block`
- Chain 0 will be processed by default – as it is now
  - New `act_chain` action to jump to another chain in the same block to be processed
  - Entrypoint is still `struct tcf_proto __rcu *filter_list` so there is no performance penalty
- Make the “block” shareable among multiple qdiscs
- Work in progress
  - https://github.com/jpirko/linux_mlxsw/commits/jiri_devel_tcmultichain

Enhancements, cleanups

Error reporting

- The infamous “We have an error talking to the kernel”
  - Great, no clue what went wrong
- Would be great to allow to pass a string message along with -ERRSOMETHING
  - How to do it?
- Ideas?

Naming consistency

- “filter” or “classifier”?
  - pkt_cls.h, tfilter_notify, struct tcf_proto, TCA_*…
- “ext” or “action”?
  - struct tcf_exts, struct tc_action, TCA_* (Different to the one mentioned above),…
- One or two letter variables (often representing totally different things)
  - t, tp, th, cl, n, b, s_t, ...
- “Namespaces” for functions, structs enums and defines
- Function names
  - something_add vs. add_something

UAPI consolidation

- Duplication of common Netlink attributes
  - TCA_BPF_ACT, TCA_FLOWER_ACT, TCA_U32_ACT, and many others
- Mixture of Netlink attributes and structure fields
- Attributes, defines and structs names
  - TC_*, TCA_*, NETEM_*, struct tc_*, struct tcf_*
- Can we do something with it?
  - Do we need a new TC UAPI? Generic netlink, parallel, deprecate the original UAPI in ~10 years?