TC removal

(okay, just kidding)

Florian Westphal

NFWS 2016, Amsterdam

June 2016

4096R/AD5FF600 fw@strlen.de
1C81 1AD5 EA8F 3047 7555
E8EE 5E2F DA6C F260 502D

nftables+egress, current state

- it doesn’t exist
- nft netdev family allows attaching nft ruleset to a device
- traversed during ingress, same as tc ingress

iptables+egress, current state

- it doesn't exist either
- `-j CLASSIFY` to set `skb->priority`
- doesn't work universally (can only select class of upper qdisc)
- Some use `-j MARK` and `fwmark` filters in tc

Current egress arch w. qdiscs/ wo bypass

- xmit routine takes qdisc root lock
- invokes `root_qdisc->enqueue(skb)`
  - qdisc enqueue function invokes `tc_classify`
  - gives the class, qdisc calls
  - `class->qdisc->enqueue` skb
  - might result in another call to `tc_classify`
    examples: HTB + PRIO + fq_codel or HFSC+DRR+codel
- qdisc unlock / dequeue op
- classification is serialized via root qdisc lock

Ugly Hack ...

Did hack to split enqueue+classify.

- xmit routine calls `root_qdisc->classify(skb, map)` before taking root qdisc lock
- classify calls classify again if needed: `class->qdisc->classify(skb, map)`
- `map`: allocated on stack, describes path through qdisc hierachy

```
struct qnode { struct qdisc *q;
void * class; }
struct map { u8 depth;
struct qnode[MAX_DEPTH]; }
```

Classification steps assign q and class for each step
Node 0 is *leaf*

Ugly Hack ... (2)

After root qdisc lock is taken:

```
q = map[0].qdisc;
err = q->enqueue(skb, q, map[0].class);
if (err...)
for (i = 1; i < map.depth; i++) {
q = map[i].qdisc;
q->notify_enqueue(skb, map[i].class);
}
```

Only leaf qdiscs implement `enqueue`
qdiscs that delegate queueing (eg. to a pfifo) implement a notify function that does needed maintenance work (e.g. mark class ready for xmit)

problems, summary

- several stats just do `foo++` → percpu counters
- must handle qdisc change or class removal during/after lockless classify
- some actions need treatment (e.g. mirred)
- did not see any showstoppers so far, police and estimators should be fine (already use locks)
- ... do you?
- do not think it makes sense to add nft egress at this time
- nft would not scale either at the moment if run w. qdisc locking
- ... so its even too early for “integrate w. qdiscs” vs ”new schedulers“ debate

Next up: some open nft issues

nftables – open issues

- keyword collisions: `uid saddr`
  - can't avoid keywords but we can't escape from allowing arbitrary strings in some places
  - can't just treat next item as literal:
    `meta uid { user, root, saddr, foo, ip }`
- back- and forward compatibility: e.g. `jump flow`
- flow is now a statement – it fails but this used to work

Time to add grammar number to output?
`type filter revision 42`
Hannes Sowa: reserve __ prefixed strings
Then use defines for new keywords

not-so-nice

- `add filter ip saddr` vs. `ip addr` – can't yet dump list of header names, even though nft has textual descriptions

open issues

- raw instructions – just support existing (libnftnl) debug output as input?
- seems like best option, extend libnftnl to parse str, have nft pass `[ some stuff ]` to libnftnl
- e.g. allow something like

  ```
  ip protocol { udp, tcp } [ payload \
     load 2b @ transport header + 2  => reg 1 ]
     == 53
  ```

  to test udp and tcp port(s) in one rule
  - need to change nft to print raw insn on output as well if deliniarization fails
  - how to handle register allocation?