NAME

flow − flow based traffic control filter

SYNOPSIS

Mapping mode:

tc filter ... **flow map key** *KEY* [*OPS*] [*OPTIONS*]

Hashing mode:

tc filter ... **flow hash keys** *KEY_LIST* [**perturb** *secs*] [*OPTIONS*]

OPS := [*OPS*] *OP*

OPTIONS := [**divisor** *NUM*] [**baseclass** *ID*] [**match** *EMATCH_TREE*] [**action** *ACTION_SPEC*]

KEY_LIST := [*KEY_LIST*] *KEY*

OP := { **or** | **and** | **xor** | **rshift** | **addend** } *NUM*

ID := *X*:*Y*

KEY := { **src** | **dst** | **proto** | **proto-src** | **proto-dst** | **iif** | **priority** | **mark** | **nfct** | **nfct-src** | **nfct-dst** | **nfct-proto-src** | **nfct-proto-dst** | **rt-classid** | **sk-uid** | **sk-gid** | **vlan-tag** | **rxhash** }

DESCRIPTION

The **flow** classifier is meant to extend the **SFQ** hashing capabilities without hard-coding new hash functions. It also allows deterministic mappings of keys to classes.

OPTIONS

action *ACTION_SPEC*
Apply an action from the generic actions framework on matching packets.

baseclass *ID*
An offset for the resulting class ID. *ID* may be **root**, **none** or a hexadecimal class ID in the form [*X***:**]*Y*. *X* must match qdisc's/class's major handle (if omitted, the correct value is chosen automatically). If the whole **baseclass** is omitted, *Y* defaults to 1.

divisor *NUM*
Number of buckets to use for sorting into. Keys are calculated modulo *NUM*.

hash keys *KEY-LIST*
Perform a **jhash2** operation over the keys in *KEY-LIST*, the result (modulo the **divisor** if given) is taken as class ID, optionally offset by the value of **baseclass**. It is possible to specify an interval (in seconds) after which **jhash2**'s entropy source is recreated using the **perturb** parameter.

map key *KEY*
Packet data identified by *KEY* is translated into class IDs to push the packet into. The value may be mangled by *OPS* before using it for the mapping. They are applied in the order listed here:

and *NUM*
Perform bitwise **AND** operation with numeric value *NUM*.

or *NUM*
Perform bitwise **OR** operation with numeric value *NUM*.

xor *NUM*
Perform bitwise **XOR** operation with numeric value *NUM*.

rshift *NUM*
Shift the value of *KEY* to the right by *NUM* bits.

addend *NUM*
Add *NUM* to the value of *KEY*.

For the **or**, **and**, **xor** and **rshift** operations, *NUM* is assumed to be an unsigned, 32bit integer value. For the **addend** operation, *NUM* may be much more complex: It may be prefixed by a minus ('-') sign to cause subtraction instead of addition and for keys of **src**, **dst**, **nfct-src** and **nfct-dst** it may be given in IP address notation. See below for an illustrating example.

match *EMATCH_TREE*
Match packets using the extended match infrastructure. See **tc-ematch**(8) for a detailed description of the allowed syntax in *EMATCH_TREE*.

KEYS

In mapping mode, a single key is used (after optional permutation) to build a class ID. The resulting ID is deducible in most cases. In hashing more, a number of keys may be specified which are then hashed and the output used as class ID. This ID is not deducible in beforehand, and may even change over time for a given flow if a **perturb** interval has been given.

The range of class IDs can be limited by the **divisor** option, which is used for a modulus.

src, **dst** Use source or destination address as key. In case of IPv4 and TIPC, this is the actual address value. For IPv6, the 128bit address is folded into a 32bit value by XOR'ing the four 32bit words. In all other cases, the kernel-internal socket address is used (after folding into 32bits on 64bit systems).

proto Use the layer four protocol number as key.

proto-src
Use the layer four source port as key. If not available, the kernel-internal socket address is used instead.

proto-dst
Use the layer four destination port as key. If not available, the associated kernel-internal dst_entry address is used after XOR'ing with the packet's layer three protocol number.

iif Use the incoming interface index as key.

priority
Use the packet's priority as key. Usually this is the IP header's DSCP/ECN value.

mark Use the netfilter **fwmark** as key.

nfct Use the associated conntrack entry address as key.

nfct-src, **nfct-dst**, **nfct-proto-src**, **nfct-proto-dst**
These are conntrack-aware variants of **src**, **dst**, **proto-src** and **proto-dst**. In case of NAT, these are basically the packet header's values before NAT was applied.

rt-classid
Use the packet's destination routing table entry's realm as key.

sk-uid
sk-gid For locally generated packets, use the user or group ID the originating socket belongs to as key.

vlan-tag
Use the packet's vlan ID as key.

rxhash Use the flow hash as key.

EXAMPLES

Classic SFQ hash:

```
tc filter add ... flow hash \
      keys src,dst,proto,proto-src,proto-dst divisor 1024
```

Classic SFQ hash, but using information from conntrack to work properly in combination with NAT:

```
tc filter add ... flow hash \
      keys nfct-src,nfct-dst,proto,nfct-proto-src,nfct-proto-dst \
      divisor 1024
```

Map destination IPs of 192.168.0.0/24 to classids 1-256:

```
tc filter add ... flow map \
      key dst addend -192.168.0.0 divisor 256
```

Alternative to the above:

```
tc filter add ... flow map \
      key dst and 0xff
```

The same, but in reverse order:

```
tc filter add ... flow map \
      key dst and 0xff xor 0xff
```

SEE ALSO

tc(8), **tc-ematch**(8), **tc-sfq**(8)