Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'tc-flower.8' command

$ man tc-flower.8

Flower filter in tc(8) Linux Flower filter in tc(8)

NAME

flower - flow based traffic control filter

SYNOPSIS

tc filter ... flower [MATCH_LIST] [action ACTION_SPEC] [classid CLASSID] [hw_tc TCID]

MATCH_LIST := [MATCH_LIST] MATCH

MATCH := { indev ifname | verbose | skip_sw | skip_hw | { dst_mac | src_mac } MASKED_LLADDR | vlan_id VID | vlan_prio PRIORITY | vlan_ethtype { ipv4 | ipv6 | ETH_TYPE } | cvlan_id VID | cvlan_prio PRIORITY | cvlan_ethtype { ipv4 | ipv6 | ETH_TYPE } | pppoe_sid PSID | ppp_proto { ip | ipv6 | mpls_uc | mpls_mc | PPP_PROTO } | mpls LSE_LIST | mpls_label LABEL | mpls_tc TC | mpls_bos BOS | mpls_ttl TTL | ip_proto { tcp | udp | sctp | icmp | icmpv6 | IP_PROTO } | ip_tos MASKED_IP_TOS | ip_ttl MASKED_IP_TTL | { dst_ip | src_ip } PREFIX | { dst_port | src_port } { MASKED_NUMBER | min_port_number-max_port_number } | tcp_flags MASKED_TCP_FLAGS | type MASKED_TYPE | code MASKED_CODE | { arp_tip | arp_sip } IPV4_PREFIX | arp_op { request | reply | OP } | { arp_tha | arp_sha } MASKED_LLADDR | enc_key_id KEY-ID | { enc_dst_ip | enc_src_ip } { ipv4_address | ipv6_address } | enc_dst_port port_number | enc_tos TOS | enc_ttl TTL | { geneve_opts | vxlan_opts | erspan_opts |

gtp_opts } OPTIONS | ip_flags IP_FLAGS }

LSE_LIST := [LSE_LIST] LSE

LSE := lse depth DEPTH { label LABEL | tc TC | bos BOS | ttl TTL }

DESCRIPTION

The flower filter matches flows to the set of keys specified and as? signs an arbitrarily chosen class ID to packets belonging to them. Ad? ditionally (or alternatively) an action from the generic action frame? work may be called.

OPTIONS

action ACTION_SPEC

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

classid CLASSID

Specify a class to pass matching packets on to. CLASSID is in the form X:Y, while X and Y are interpreted as numbers in hexa? decimal format.

hw_tc TCID

Specify a hardware traffic class to pass matching packets on to. TCID is in the range 0 through 15.

indev ifname

Match on incoming interface name. Obviously this makes sense only for forwarded flows. ifname is the name of an interface which must exist at the time of tc invocation.

verbose

Enable verbose logging, including offloading errors when not us? ing skip_sw flag.

skip_sw

Do not process filter by software. If hardware has no offload support for this filter, or TC offload is not enabled for the interface, operation will fail.

skip_hw

Do not process filter by hardware.

dst_mac MASKED_LLADDR

src_mac MASKED_LLADDR

Match on source or destination MAC address. A mask may be optionally provided to limit the bits of the address which are matched. A mask is provided by following the address with a slash and then the mask. It may be provided in LLADDR format, in which case it is a bitwise mask, or as a number of high bits to match. If the mask is missing then a match on all bits is assumed.

num_of_vlans NUM

Match on the number of vlan tags in the packet. NUM can be 0 or small positive integer. Typically in 0-4 range.

vlan_id VID

Match on vlan tag id. VID is an unsigned 12bit value in decimal format.

vlan_prio PRIORITY

Match on vlan tag priority. PRIORITY is an unsigned 3bit value in decimal format.

vlan_ethtype VLAN_ETH_TYPE

Match on layer three protocol. VLAN_ETH_TYPE may be either ipv4, ipv6 or an unsigned 16bit value in hexadecimal format. To match on QinQ packet, it must be 802.1Q or 802.1AD.

cvlan_id VID

Match on QinQ inner vlan tag id. VID is an unsigned 12bit value in decimal format.

cvlan_prio PRIORITY

Match on QinQ inner vlan tag priority. PRIORITY is an unsigned 3bit value in decimal format.

cvlan_ethtype VLAN_ETH_TYPE

Match on QinQ layer three protocol. VLAN_ETH_TYPE may be either ipv4, ipv6 or an unsigned 16bit value in hexadecimal format.

pppoe_sid PSID

Match on PPPoE session id. PSID is an unsigned 16bit value in decimal format.

ppp_proto PPP_PROTO

Match on PPP layer three protocol. PPP_PROTO may be either ip, ipv6, mpls_uc, mpls_mc or an unsigned 16bit value in hexadecimal format.

mpls LSE_LIST

Match on the MPLS label stack. LSE_LIST is a list of Label Stack Entries, each introduced by the lse keyword. This option can't be used together with the standalone mpls_label, mpls_tc, mpls_bos and mpls_ttl options.

lse LSE_OPTIONS

Match on an MPLS Label Stack Entry. LSE_OPTIONS is a list of options that describe the properties of the LSE to match.

depth DEPTH

The depth of the Label Stack Entry to consider. Depth starts at 1 (the outermost Label Stack En? try). The maximum usable depth may be limited by the kernel. This option is mandatory. DEPTH is an unsigned 8 bit value in decimal format.

label LABEL

Match on the MPLS Label field at the specified depth. LABEL is an unsigned 20 bit value in deci? mal format.

tc TC Match on the MPLS Traffic Class field at the spec? ified depth. TC is an unsigned 3 bit value in decimal format.

bos BOS

Match on the MPLS Bottom Of Stack field at the specified depth. BOS is a 1 bit value in decimal format.

ttl TTL

Match on the MPLS Time To Live field at the speci? fied depth. TTL is an unsigned 8 bit value in

decimal format.

mpls_label LABEL

Match the label id in the outermost MPLS label stack entry. LA? BEL is an unsigned 20 bit value in decimal format.

mpls_tc TC

Match on the MPLS TC field, which is typically used for packet priority, in the outermost MPLS label stack entry. TC is an un? signed 3 bit value in decimal format.

mpls_bos BOS

Match on the MPLS Bottom Of Stack field in the outermost MPLS label stack entry. BOS is a 1 bit value in decimal format.

mpls_ttl TTL

Match on the MPLS Time To Live field in the outermost MPLS label stack entry. TTL is an unsigned 8 bit value in decimal format.

ip_proto IP_PROTO

Match on layer four protocol. IP_PROTO may be tcp, udp, sctp, icmp, icmpv6 or an unsigned 8bit value in hexadecimal format.

ip_tos MASKED_IP_TOS

Match on ipv4 TOS or ipv6 traffic-class - eight bits in hexadec? imal format. A mask may be optionally provided to limit the bits which are matched. A mask is provided by following the value with a slash and then the mask. If the mask is missing then a match on all bits is assumed.

ip_ttl MASKED_IP_TTL

Match on ipv4 TTL or ipv6 hop-limit - eight bits value in deci? mal or hexadecimal format. A mask may be optionally provided to limit the bits which are matched. Same logic is used for the mask as with matching on ip_tos.

dst_ip PREFIX

src_ip PREFIX

Match on source or destination IP address. PREFIX must be a valid IPv4 or IPv6 address, depending on the protocol option to tc filter, optionally followed by a slash and the prefix length.

If the prefix is missing, tc assumes a full-length host match.

dst_port { MASKED_NUMBER | MIN_VALUE-MAX_VALUE }

src_port { MASKED_NUMBER | MIN_VALUE-MAX_VALUE }

Match on layer 4 protocol source or destination port number, with an optional mask. Alternatively, the minimum and maximum values can be specified to match on a range of layer 4 protocol source or destination port numbers. Only available for ip_proto values udp, tcp and sctp which have to be specified in before? hand.

tcp_flags MASKED_TCP_FLAGS

Match on TCP flags represented as 12bit bitfield in in hexadeci? mal format. A mask may be optionally provided to limit the bits which are matched. A mask is provided by following the value with a slash and then the mask. If the mask is missing then a match on all bits is assumed.

type MASKED_TYPE

code MASKED_CODE

Match on ICMP type or code. A mask may be optionally provided to limit the bits of the address which are matched. A mask is pro? vided by following the address with a slash and then the mask. The mask must be as a number which represents a bitwise mask If the mask is missing then a match on all bits is assumed. Only available for ip_proto values icmp and icmpv6 which have to be specified in beforehand.

arp_tip IPV4_PREFIX

arp_sip IPV4_PREFIX

Match on ARP or RARP sender or target IP address. IPV4_PREFIX must be a valid IPv4 address optionally followed by a slash and the prefix length. If the prefix is missing, tc assumes a full-length host match.

arp_op ARP_OP

Match on ARP or RARP operation. ARP_OP may be request, reply or an integer value 0, 1 or 2. A mask may be optionally provided

to limit the bits of the operation which are matched. A mask is provided by following the address with a slash and then the mask. It may be provided as an unsigned 8 bit value representing a bitwise mask. If the mask is missing then a match on all bits is assumed.

arp_sha MASKED_LLADDR

arp_tha MASKED_LLADDR

Match on ARP or RARP sender or target MAC address. A mask may be optionally provided to limit the bits of the address which are matched. A mask is provided by following the address with a slash and then the mask. It may be provided in LLADDR format, in which case it is a bitwise mask, or as a number of high bits to match. If the mask is missing then a match on all bits is as? sumed.

enc_key_id NUMBER

enc_dst_ip PREFIX

enc_src_ip PREFIX

enc_dst_port NUMBER

enc_tos NUMBER

enc_ttl NUMBER

ct_state CT_STATE

ct_zone CT_MASKED_ZONE

ct_mark CT_MASKED_MARK

ct_label CT_MASKED_LABEL

Matches on connection tracking info

CT_STATE

Match the connection state, and can be combination of [{+|-}flag] flags, where flag can be one of

trk - Tracked connection.

new - New connection.

est - Established connection.

rpl - The packet is in the reply direction, meaning that it is in the opposite direction from the packet that ini?

tiated the connection.

inv - The state is invalid. The packet couldn't be asso?

ciated to a connection.

rel - The packet is related to an existing connection.

Example: +trk+est

CT_MASKED_ZONE

Match the connection zone, and can be masked.

CT_MASKED_MARK

32bit match on the connection mark, and can be masked.

CT_MASKED_LABEL

128bit match on the connection label, and can be masked.

geneve_opts OPTIONS

vxlan_opts OPTIONS

erspan_opts OPTIONS

gtp_opts OPTIONS

Match on IP tunnel metadata. Key id NUMBER is a 32 bit tunnel key id (e.g. VNI for VXLAN tunnel). PREFIX must be a valid IPv4 or IPv6 address optionally followed by a slash and the prefix length. If the prefix is missing, tc assumes a full-length host match. Dst port NUMBER is a 16 bit UDP dst port. Tos NUMBER is an 8 bit tos (dscp+ecn) value, ttl NUMBER is an 8 bit time-to-live value. geneve_opts OPTIONS must be a valid list of comma-separated geneve options where each option consists of a key op? tionally followed by a slash and corresponding mask. If the masks is missing, tc assumes a full-length match. The options can be described in the form CLASS:TYPE:DATA/CLASS_MASK:TYPE_MASK:DATA_MASK, where CLASS is represented as a 16bit hexadecimal value, TYPE as an 8bit hexa? decimal value and DATA as a variable length hexadecimal value. vxlan_opts OPTIONS doesn't support multiple options, and it con? sists of a key followed by a slash and corresponding mask. If the mask is missing, tc assumes a full-length match. The option can be described in the form GBP/GBP_MASK, where GBP is repre?

sented as a 32bit number. erspan_opts OPTIONS doesn't support multiple options, and it consists of a key followed by a slash and corresponding mask. If the mask is missing, tc assumes a full-length match. The option can be described in the form VER? SION:INDEX:DIR:HWID/VERSION:INDEX_MASK:DIR_MASK:HWID_MASK, where VERSION is represented as a 8bit number, INDEX as an 32bit num? ber, DIR and HWID as a 8bit number. Multiple options is not sup? ported. Note INDEX/INDEX_MASK is used when VERSION is 1, and DIR/DIR_MASK and HWID/HWID_MASK are used when VERSION is 2. gtp_opts OPTIONS doesn't support multiple options, and it con? sists of a key followed by a slash and corresponding mask. If the mask is missing, tc assumes a full-length match. The option can be described in the form PDU_TYPE:QFI/PDU_TYPE_MASK:QFI_MASK where both PDU_TYPE and QFI are represented as a 8bit hexadeci? mal values.

ip_flags IP_FLAGS

IP_FLAGS may be either frag, nofrag, firstfrag or nofirstfrag where frag and nofrag could be used to match on fragmented pack? ets or not, respectively. firstfrag and nofirstfrag can be used to further distinguish fragmented packet. firstfrag can be used to indicate the first fragmented packet. nofirstfrag can be used to indicates subsequent fragmented packets or non-fragmented packets.

NOTES

As stated above where applicable, matches of a certain layer implicitly depend on the matches of the next lower layer. Precisely, layer one and two matches (indev, dst_mac and src_mac) have no dependency, MPLS and layer three matches (mpls, mpls_label, mpls_tc, mpls_bos, mpls_ttl, ip_proto, dst_ip, src_ip, arp_tip, arp_sip, arp_op, arp_tha, arp_sha and ip_flags) depend on the protocol option of tc filter, layer four port matches (dst_port and src_port) depend on ip_proto being set to tcp, udp or sctp, and finally ICMP matches (code and type) depend on ip_proto being set to icmp or icmpv6.

There can be only used one mask per one prio. If user needs to specify different mask, he has to use different prio.

SEE ALSO

tc(8), tc-flow(8)