Extend TC to support Connection Tracking

Guy Shattah, Rony Efraim
Mellanox, Ra’anana, Israel
[sguy | ronye] at mellanox.com

Abstract

Recent industry movement towards the use of TC as well as Open vSwitch (OVS) has created new requirements, one of which is supporting connection tracking (CT).
OVS data-path already uses rules to deliver the packet to CT and rules to act according to CT State.
TC should also be extended to support rules according to CT State. In this paper, we would like to present a suggestion on how TC API should be extended in order to use CT as an action and the ability to classify on connection state.

Keywords

Virtualization, switchdev, TC, filters, Open vSwitch, Software Defined Networking, connection tracking

Introduction

Connection tracking

Stateful traffic filters allow finer-grained traffic analysis which provide richer information that allows sysadmins and security experts to define more intelligent policies. Linux Netfilter Connection Tracking provides such state semantics. The connection tracking subsystem stores information about the state of a connection in a memory structure that contains the source and destination IP addresses, port number pairs, pro- tocol types, state, and timeout. With this extra information, more intelligent filtering policies can be defined.[1]

Open vSswitch

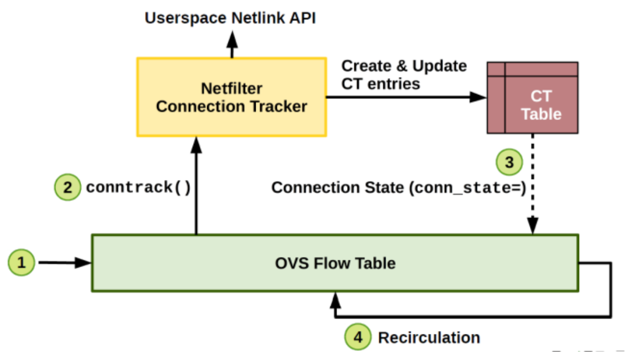
OVS (Open Virtual Switch) is one of the dominant virtual switches, it is capable of switching frames between local VMs on the host (sometimes called east-west traffic) and between local VMs and remote VMs (sometimes called north-south traffic). One major difference between OVS and a “regular” IEEE Ethernet bridge is that the OVS switch “flows” as oppose to a regular Ethernet bridge which provides frame delivery between VMs based on MAC/VLAN. [2]

OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network. It enables network controllers to determine the path of network packets across a network of switches, while the controllers are distinct from the switches. [3]

OVS supports making decisions based on connection tracking by using OpenFlow Commands

```
table=0,priority=100,ip,ct_state=-trk,action=ct(table=1)
table=1,in_port=1,ip,ct_state=+trk+new,action=ct(commit),2
table=1,in_port=1,ip,ct_state=+trk+est,action=2
table=1,in_port=2,ip,ct_state=+trk+new,action=drop
table=1,in_port=2,ip,ct_state=+trk+est,action=1
```

It start at table 0, any incoming IP packet with missing connection state data (ct_state = - trk) is sent to connection-tracking and from there to table: 1. Where packets from port 1 (in_port = 1) are tested for connection-state. The tracked ones (+trk) which are part of a new connection (+new) are being committed (commit) and output to port 2. Tracked ones (+trk) which are part of established connection (+est) are sent directly to port 2. Packets from port2 that are tracked (+trk) which are part of a new connection (+new) are dropped, and the established connection (+est) are sent directly to port 1.


Userspace Netlink API
Netfilter Connection Tracker
Create & Update CT entries
CT Table
3
2 conntrack()
Connection State (conn_state=)
1
OVS Flow Table
4 Recirculation

TC

TC is Linux Traffic Control mechanism that includes the sets of queuing systems and mechanisms by which packets are received (via Ingress port) and transmitted (via Egress port) on a router. [4]

Classifiers/Filters are selectors of packets in **TC**. They stare at either packet data or meta-data and select an action to execute. Each classifier type implements its own algorithm and is specialized. A classifier contains filters which implement semantics applicable to the classifier algorithm. For each policy defined, there is a built in filter which matches first based on the layer 2 protocol type. **Actions** are executed when a resulting classifier filter matches. [5]

While other classifiers could be used, The goal of this paper is to focus on extending the flower classifier. **The flower classifier** was written by Jiří Pírko and makes use of several "commodity" kernel features. As a packet traverses the stack, the flow it uses is cached. Flower then stores the flow in a rhashtable. Subsequent packets matching a classifier rule can quickly be directed into the correct flow by retrieving the cached copy from the rhashtable. Flower supports rules based on a subset of possible flow parameters (source and destination address, ingress and egress ports, MAC addresses, etc) and more. [6]

The flower classifier supports action-chaining with Multi-table/Multi-chain: TC rules (filters) are put together into chains in order according to priority (pref). Each chain can be looked at as a table of rules in order to decide of an action. An action is an order to execute when a resulting classifier filter matches. Below is an example for a chain.

Insert a rule into specific chain:
$ tc filter add dev eth1 parent ffff: protocol ip **chain 100** pref 10 flower dst_ip 192.168.101.1 action drop

Use “goto chain” action:
$ tc filter add dev eth1 parent ffff: protocol ip pref 10 flower src_ip 192.168.101.1 action **goto chain 100**

Tc also has the connmark action, connmark provides a way to have a mark which is linked to a connection tracking entry. Despite the similar sounding name, connmark does not act as a classifier (it is an action) nor does it allow to act based on connection state.

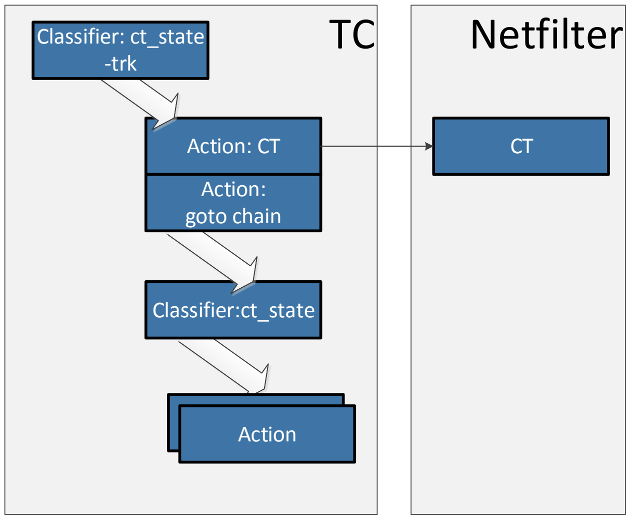
Motivation

Iptables, also known as netfilter, is a command-line firewall utility that uses policy chains to allow or block traffic. Connection-tracking is a part of the netfilter subsystem. Hence iptables is capable of making decisions whether to accept, reject or drop packets based on the connection state. Being a stand-alone subsystem, it is possible to easily integrate Connection tracking as part of TC, Hence adding options (which did not exist before) to filter traffic based on connection state.

Recent hardware devices allow offloaded connection tracking. OVS, being a customer of the connection tracking subsystem, could benefit from CT offloading. OVS is already capable of using TC both for classification and hardware offload [7]. Utilizing the same hardware to offload twice on the very same data path would introduce a performance hit. Hence we would like to offload it once. Offloading both TC and CT, is possible by integrating CT inside TC thus allowing TC to perform all the classification.

TC suggestion

OVS supports connection tracking by utilizing the kernel CT module, We suggest to do the same by add CT to TC as an action and match. New CT action, send the packet to CT, and continue to the requested chain (table). Where a decision will be made based (not limiting to) on match the connection state.



New action “ct” will be add in order to call the nf_ct.
The new ct action can have the flowing optional parameters:

- commit - commit the connection
- zone <number> - Zone number in CT to use (u16)

New match will be add to flower classifier call ct_state, to classify the connection state. ct_state flags can be match to be set by using “+” or clear by using “-“, all other flags will be ignored. The flags are:

- <u>trk</u> - Tracked - Been through the connection tracker
- <u>inv</u> – Invalid
- <u>new</u> – new connection
- <u>est</u> - Established connection
- <u>rpl</u> - Packet is in reply direction
- <u>rel</u> - Related - ICMP, eg “dst_unreach” response or helper “related” connection

Below there is example for use the TC for CT rules:

```
tc filter add dev eth5 protocol ip parent ffff: chain 0
  flower ct_state -trk
  action ct
  action goto chain 1

tc filter add dev eth6 protocol ip parent ffff: chain 0
  flower ct_state -trk
  action ct
  action goto chain 2

tc filter add dev eth5 protocol ip parent ffff: chain 1
  flower ct_state +trk,+est
  action mirred egress redirect dev eth6

tc filter add dev eth6 protocol ip parent ffff: chain 2
  flower ct_state +trk,+est
  action mirred egress redirect dev eth5
```

Conclusion

The suggested changes add connection tracking offload to TC. Linking between TC and the existing Linux sub-system, Hence enabling improved performance when using hardware offload.

References

[1] Pablo Neira Ayuso, “Netfilter’s connection tracking system”, :login; the USENIX magazine. JUNE 2006

[2] Rony Efraim, Or Gerlitz, “Using SR-IOV offloads with Open-vSwitch and similar applications”, Proceedings of Netdev 1.2, Feb 2017

[3] Nick McKeown; et al. (April 2008). "OpenFlow: Enabling innovation in campus networks". ACM Communications Review.

[4] Martin A. Brown, “The Linux documentation project”. http://tldp.org/HOWTO/Traffic-Control-HOWTO/overview.html

[5] Jamal Hadi Salim – “Linux Traffic Control Classifier-Action Subsystem Architecture”, Proceedings of Netdev 0.1, Feb 2015

[6] Nathan Willis - “Measuring packet classifier performance”, https://lwn.net/Articles/675056/

[7] Jiří Pírko – “Implementing Open vSwitch datapath using TC”, Proceedings of Netdev 0.1, Feb 2015