Master thesis project: Integrating cBPF into eBPF for packet matching

BPF is a technology that allows injecting bits of code into the Linux kernel for various purposes. One of them is for processing network packets, enabling the building of fully programmable data planes. (See https://ebpf.io/what-is-ebpf for an introduction to BPF)

While most of the development of BPF is in the extended 'eBPF' variant, the 'classical BPF' (or cBPF) variant is still used to express packet matching filters in software like libpcap (used by, e.g., tcpdump). However, using such cBPF filters from eBPF programs is cumbersome: there exists a userspace conversion tool[1], but it requires recompilation of the resulting eBPF program.

The goal of this project is to improve upon this situation. The kernel already contains a facility to convert a cBPF filter program to an eBPF program that can be JIT-compiled and run as a TC filter. The task here is to extend that facility so that a cBPF filter can be evaluated using a eBPF helper and the result returned to the caller. A rough outline of the project is as follows:

- Implement a new map type that will hold cbpf byte code programs in each map item for use from TC eBPF and XDP.
- Extend the existing conversion and JIT tool to be able to compile the bytecode as it is inserted into this map
- Implement a new helper to call the cBPF programs and return the evaluation result.

All of the tasks above involve kernel programming, and a willingness to dive into the particulars of the different BPF byte code formats and instruction sets. Performance evaluation and validation of the prototype is an important part of the project.

The thesis project is offered in collaboration with Red Hat Research.

KaU contact: Anna Brunstrom

[1]https://github.com/cloudflare/cbpfc