Packet Marking With eBPF

Tristan Sullivan
University of Victoria
May 5/22

What is eBPF?

- Extended Berkeley Packet Filter; exists since about 2015 (kernel 4.4). Used for tracing and networking
- Definition from docs.cilium.io/en/stable/bpf:

  “BPF is a general purpose RISC instruction set and was originally designed for the purpose of writing programs in a subset of C which can be compiled into BPF instructions through a compiler back end (e.g. LLVM), so that the kernel can later on map them through an in-kernel JIT compiler into native opcodes for optimal execution performance inside the kernel.”
- eBPF programs execute when certain kernel hooks occur
  - xdp: as soon as network driver receives a packet (ingress only)
  - tc: later in the network stack (ingress and egress)
- The skb struct is passed to the eBPF program; gives full control over every packet

Bcc

- There exists a project to make eBPF useable via a mixture of python and C: bcc

- Writing and loading eBPF programs with bcc is pretty easy, and there are plentiful examples: https://github.com/iovisor/bcc/

- Package for RHEL-like OS is called python3-bcc, available on RHEL 8+ and friends (RHEL 7 has python-bcc, which is python2 based)

Packet Marking

- Usability with python makes it easy to turn this into a flowd backend:
  - flowd plugin gets notified of a flow that is to be marked, passes the flow identifier to the backend
  - The backend then updates a data structure that is passed between the python code and the eBPF program; it is a hash with IPv6 address as the key and flow label as the value
  - The eBPF program checks if the destination IP of each IPv6 packet is a key in the hash, and if so, puts the flow label from the hash onto the packet

Status

- Not fully integrated with flowd yet, but I have the python code to fill a data structure with IPv6 addresses pointing to flow labels, and the eBPF code to mark packets with those flow labels: https://github.com/hep-gc/Packet-Marking

- Next step is to test the impact of the eBPF program on performance. There was a 100G testbed set up by Starlight and Compute Canada for SC21 that still exists; I’ve successfully run my eBPF program on the nodes, will do iperf tests in the coming days