William (Cheng-Chun) Tu
VMware
OVS Conference 2016
Offloading OVS Flow Processing using eBPF
What is eBPF?

- A way to write a **restricted C** program and runs in **Linux kernel**
  - A new instruction set, but no corresponding HW
  - A virtual machine running in Linux kernel
  - Safety guaranteed by BPF verifier

- Maps
  - Efficient key/value store resides in kernel space
  - Can be shared between eBPF programs and user space applications

- Helper Functions
  - A core kernel defined set of functions for eBPF program to retrieve/push data from/to the kernel
Motivation

• Extensibility, when introducing a new datapath feature:
  • Upstream process provides valuable feedbacks
  • Time to upstream could also be unpredictable
  • Maintain ABI compatibility between different kernel and OVS versions.

• Maintenance cost and compatibility effort
  • Keep up with new kernel API changes
  • Backport new features to older version
  • Bugs in compat code are easy to introduce and often non-obvious to fix

• eBPF: Implement datapath functionalities in eBPF and reduce dependencies on different kernel versions
Review: Linux and OVS Datapath

Current OVS DP:
- Receive packets from bridge/device hook
- Parse -> Lookup -> Actions
Introduce eBPF actions:
• Add BPF hook point in OVS kernel DP for actions
• New actions could be added without updating OVS kernel module

Limitations:
• Parsing new protocols
• Matching new fields
OVS eBPF Datapath

**Goal**
- Replace OVS kernel datapath entirely with eBPF
- ovs-vswitchd controls and manages the eBPF DP
- eBPF map as channels in between
Agenda

• Header Parsing
• Flow Table Lookup
• Action Execution
• Performance Evaluation
Parsing Headers/Metadata using P4

P4-to-eBPF
- Leverage P4-to-eBPF compiler from bcc
- Generate protocol/metadata headers
- Parser walks through the protocol parsing graph
- Deparser writes back the packet changes
- Maps for flow lookup and counters

Limitations for OVS:
- OVS requires Linux-specific metadata fields
- OVS implements its own match + action eBPF program
Review: Flow Lookup in kernel Datapath

Slow Path:
- Ingress: lookup miss and upcall
- `ovs-vswitchd` receives, does flow translation, and programs flow entry into flow table in OVS kernel module
- OVS kernel DP installs the flow entry
- OVS kernel DP receives and executes actions on the packet

Fast Path:
- Subsequent packets hit the flow cache
Flow Lookup in eBPF Datapath

Slow Path:
- Ingress: lookup miss and upcall
- Perf ring buffer carries packet and its metadata to ovs-vswitchd
- ovs-vswitchd receives, does flow translation, and programs flow entry into eBPF map
- ovs-vswitchd sends the packet down to trigger lookup again

Benefits:
- Use any fixed binary format between userspace and kernel eBPF program.

Limitation at flow installation:
TLV format currently not supported in BPF verifier
Solution: Convert TLV into fixed length array
Review: OVS Kernel Datapath Actions

A list of actions to execute on the packet

Example cases of DP actions

- Flooding:
  - Datapath actions: 9,55,10,55,66,11,77,88,9,1

- Mirror and push vlan:
  - Datapath actions: 3,push_vlan(vid=17,pcp=0),2

- Tunnel:
  - Datapath actions:
    set(tunnel(tun_id=0x5,src=2.2.2.2,dst=1.1.1.1,ttl=64,flags(df|key))),1
eBPF Datapath Actions

A list of actions to execute on the packet

Challenges
• Limited eBPF program size (maximum 4K instructions)
• Variable number of actions: BPF disallows loops to ensure program termination

Solution:
• Make each action type an eBPF program, and tail call the next action
• Side effects: tail call has limited context and does not return
• Solution: keep action metadata and action list in a map
Performance Evaluation

- Sender sends 64Byte, 14.88Mpps to one port, measure the receiving packet rate at the other port
- OVS receives packets from one port, forwards to the other port
- Compare OVS kernel datapath and eBPF datapath
- Measure **single flow, single core** performance with Linux kernel 4.9-rc3 on OVS server
OVS Kernel and eBPF Datapath Performance

Recap: eBPF DataPath

- tc ingress
- Parser
- Lookup
- Actions
- deparser
- tc egress

### eBPF DP Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Mpps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redirect (no parser, lookup, actions, deparser)</td>
<td>1.90</td>
</tr>
<tr>
<td>Hash</td>
<td>1.12</td>
</tr>
<tr>
<td>Push vlan</td>
<td>1.11</td>
</tr>
<tr>
<td>Set dst_mac</td>
<td>0.84</td>
</tr>
<tr>
<td>Set dst_mac &lt;no deparser&gt;</td>
<td>1.14</td>
</tr>
<tr>
<td>Set GRE tunnel</td>
<td>0.48</td>
</tr>
</tbody>
</table>

All measurements are based on single flow, single core.

### OVS Kernel DP Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Mpps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>1.34</td>
</tr>
<tr>
<td>Set dst_mac</td>
<td>1.23</td>
</tr>
<tr>
<td>Set GRE tunnel</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Opportunity for improving parser and deparser
Conclusion and Future Work

Conclusion
• Feasibility of implementing OVS Datapath entirely using eBPF
• Decouple OVS datapath functionality from kernel versions
• Limitation of eBPF might incur performance overhead

Future work
• Complete all the datapath actions, ex: connection tracking
• Megaflow lookup using eBPF map
Thank You

Question?
Twitter: @u9012063
u9012063@gmail.com