A Control & Management Plane for IOModules

Mauricio Vásquez Bernal, Matteo Bertrone, Sebastiano Miano, Fulvio Risso
Politecnico di Torino, Italy
Why IOVisor for NFV?

Because IOModules are fast...

L2 switch, 32 MAC addrs, IOVisor vs OVS (Mpps)

UDP packets sent to 32 different MAC addresses
Why IOVisor for NFV?

...and become even **faster** in case of chains, without even considering **XDP**

<table>
<thead>
<tr>
<th></th>
<th>TCP Throughput (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOV Namespaces</td>
<td>57,1</td>
</tr>
<tr>
<td>IOV VM</td>
<td>36,16</td>
</tr>
<tr>
<td>OpenStack Vanilla</td>
<td>23,74</td>
</tr>
</tbody>
</table>

sender – L2 Switch – Router – L2 Switch – receiver
Why IOVisor for NFV?

Because IOModules are **flexible**
Why IOVisor for NFV?

Because IOModules help to create **more resilient** services and provide **low latency** operations.

Host1 -> Router IOModule

ARP Request

ARP Reply

Host1

ARP Request

ARP Request
Why IOVisor for NFV?

Because IOModules are portable and consume a few resources.

MIPS/ARM CPU
~128MB RAM
50-100 USD

Intel x64CPU
~64-128GB RAM
5000 USD

50-100 USD
Why IOVisor for NFV?

Because IOModules are powerful

Really so powerful?

- L2 switch
- DHCP
- Router
- Network Address and Port Translator
- One-to-one NAT
- DPI
In fact, VNF IOModules are missing a standard **slow path**

- **Why a slow path?**
  - eBPF virtual machine is limited (e.g., not Turing complete, max number of instructions)
  - Complex VNFs cannot be implemented

- **L2 Switch**
  - Small and fixed number of phy ports: broadcast is implemented with a unrolled loop
  - Cannot broadcast a packet to multiple IOModules (tail call)

- **Router**
  - Cannot enqueue packets while issuing an ARP request
  - Difficult to add ICMP support (e.g., replying to Echo Requests, generation of TTL Exceeded)

- **DHCP**
  - Maximum address pool size is 10 (lack of a FIFO-like map)
  - Difficult to support some characteristics of the protocol (e.g., address rebinding and releasing)
Proposal: split IOModules in two components

- **Fast path** implemented in kernel using eBPF
- **Slow path** implemented in user-space
  - The user-space software includes also Management and Control
- Not necessarily on the same server
Fast and Slow Path

- Fast path is able to send packets and their metadata to the slow path
  - Similar to Packet-In in OpenFlow
- A packet is sent to the slow path when:
  - Fast path does not know how to handle the packet
    - E.g.: Missing entry in ARP table
    - It is convenient to process that packet in user-space
      - E.g.: DHCP messages
- Slow path is able to send packets to the fast path
  - Same as Packet-Out in OpenFlow
Architecture – Control Plane

- Each IOModule (slow path) `Register()` a callback with a specific IOModuleID
- **Controller**: when it receives the packet, it decides which is the destination (slow path) IOModule
- **Packet-in**: A new packet for an IOModule Slowpath has arrived
- **Packet-out**: Send this packet to an IOModule interface, Egress or Ingress
Architecture - Dataplane

- **Local Controller**: 
  - Polls packets from a *Perf Ring Buffer* and sends them to the controller
  - Receives packets from the controller and sends them to the decapsulator using a tap interface

- **Encapsulator**: receives a packet from an IOModule, collects its metadata and sends it to the local controller using a Perf Ring buffer

- **Decapsulator**: receives a packet from the local controller, reads the metadata from a map and forwards the packet to the proper IOModule
Implementation details

- IOModule -> Local Controller
  - Two options
    1. Use a Perf Ring Buffer (`perf_submit_skb()` function) (*current implementation*)
    2. Send the packet through a TAP interface and save the metadata in a map

- Local Controller -> IOModule
  - Packets are transferred using a TAP interface
  - Metadata are saved in an array map
    - Currently a producer-consumer mechanism between decapsulator and Local Controller guarantees the relationship between metadata and packet
    - We are exploring another approaches like pushing a tag to the packet and using it as a key in the metadata map (to avoid loss of synchronism)
  - Is there a mechanism like a Perf Ring Buffer from user to kernel?
User Space Helpers

- The local controller act also as a sort of “user space helper”
  - Placing there generic features that could be useful for different IOModules
    - E.g., broadcast a packet:
      - useful to broadcast a packet to more than one IOModule
      - not allowed with the vanilla implementation due to the tail call
  - Implemented within the local controller, packets do not travel over a TCP connection
    - Faster (processed on the host), while we do not have to touch the kernel
Results – Switch

- Removed the limitation related to the maximum number of physical ports
  - Was 32, now it could be the max number of entries in a hash map
    - Was limited to 32 because an unrolled loop was needed to perform broadcast

- Packet can be broadcasted multiple IOModules
  - It was only possible to broadcast it to a single IOModule because links between IOModules are implemented using tailcalls

- Code is much simpler and clean
  - There were many workarounds to overcome the broadcast limitations
Results – Switch and code cleanup (1)
Results – Switch and code cleanup (2)

Author: Mauricio Vasquez B <mauricio.vasquez@polito.it>
Date: Thu Feb 16 15:13:12 2017 -0500

iomodules/switch: use broadcast action

Given that the broadcast is now implemented in the local controller, all the broadcast logic implemented on the switch can be removed.

iomodules/l2switch/switch.go | 49 +----------------------
iomodules/l2switch/switchAPI.go | 99 ++++++++++++++++++++++++
2 files changed, 22 insertions(+), 126 deletions(-)

Removed an high number of lines in the ctrl/mgmt plane as well
Results – Router

• Packets are not lost when an entry in the ARP table is not found
  – If ARP table miss then packet is sent to slowpath, which saves the packet in a buffer and sends the ARP request
  – When the ARP Reply is received, buffered packets are sent out
• We can also add support for more complex protocols (e.g., ICMP)

```c
#DATAPATH

/*...*/
// ARP Lookup MISS for next_hop
pkt_set_metadata(skb, &mdata);
pkt_controller(skb, md, SLOWPATH_ARP_LOOKUP_MISS);
return RX_CONTROLLER;

/*...*/
// notify the slowpath. New arp reply received.
pkt_controller(skb, md, SLOWPATH_ARP_REPLY);
return RX_CONTROLLER;
```

```c
#SLOWPATH

switch packet.Reason {
    case SLOWPATH_ARP_LOOKUP_MISS:
        // Enqueue packet into a buffer
        // Send out ARP Request
    case SLOWPATH_ARP_REPLY:
        // Send packet(s) from buffer to next_hop
        // destination
}
```
Results - DHCP

• Server logic implemented in the slow path
  – The full implementation of the protocol is now supported
    • No longer limited to a pool size of 10 addresses
    • Added support for address rebinding and releasing

• Fast path performs a filtering on the packets
  – Only DHCP packets are sent to the controller

• A smarter data-plane DHCP is planned
Status of the IOVisor-OVN project

- Need to restart coding...
  - We needed IOModules that were not available at that time

- This is still under the same repo (iovisor-ovn)
  - [https://github.com/iovisor/iovisor-ovn](https://github.com/iovisor/iovisor-ovn)
  - Now under IOVisor github

- In fact, all the control plane has been implemented in the same repo

- Added a “standalone” working mode to the `iovisor-ovn` daemon which allow to play just with chains of IOModules, without attaching to OVN
Next Steps

• Specs and API review (on public iovisor mlist)
  – Integrate with lib iov?

• Currently a special purpose controller, written in Go: move to a generic SDN controller (e.g., ONOS)?

• Scientific dissemination
  – E.g., http://opennetworking.kr/ossn: “Authors are invited to submit papers that fall into any topics related with open-source software networking. All papers accepted for the presentation on the workshop MUST refer to available open-source implementations, which must be cited in the paper and publicly available at the time of the review.”

• Share development effort