Virtual Open Systems

A novel pflua-based OpenFLow implementation for VOSYSwitch

contact@virtualopensystems.com

April 2018
IEEE FMEC 2018, Barcelona

Authorship and sponsorship

Jeremy FANGUÈDE, Virtualization and Embedded Systems engineer at Virtual Open Systems (VOSYS).

Virtual Open Systems is a high-tech software company active in open source virtualization solutions and custom services for complex mixed-criticality automotive systems, NFV networking infrastructures, consumer electronics, mobile devices and in general for embedded heterogeneous multicore systems around new generation processor architectures.

This work is done in the context of the H2020 “Distributed Cloud & Radio Platform for 5G Neural Hosts” project (www.5gcity.eu).

Agenda

- Software switch and OpenFlowWhy software switch? Why OpenFlow?
- VOSYSwitch
- PF-Lua OpenFlow implementation
- Benchmark configuration
- Benchmark results

Edge computing and network switches

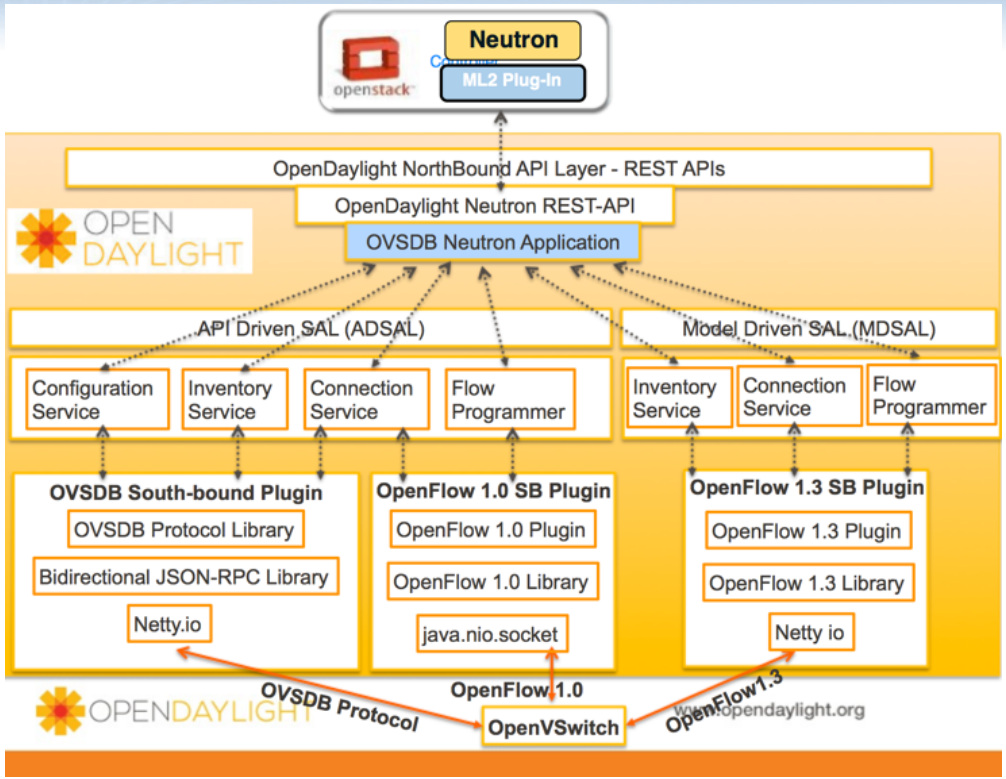
- Software Defined Network (SDN) and Network Function Virtualization (NFV) technologies are emerging in the Edge Computing
- Virtual switches are a key component of SDN/NFV infrastructures
- They provides much more flexible network architecture

Software Defined Networking (SDN)

SDN is an innovatory approach for cloud computing that facilitates network management and enables efficient network configuration in order to improve network performance and monitoring.

SDN is disassociated into two main planes:

- **Data-plane** – forwarding process of network packets
- **Control-plane** – routine process – SDN controller is configuring the switch/bridge through **OpenFlow** or other SDN protocol.

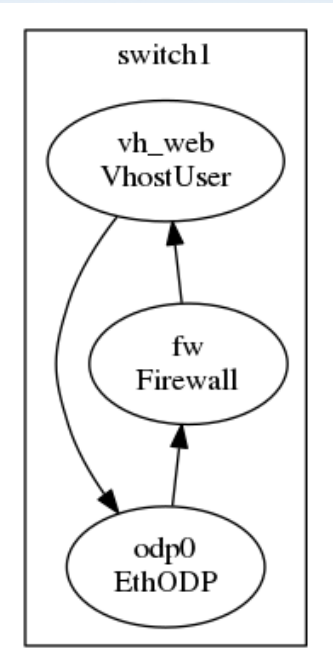


What is OpenFlow?

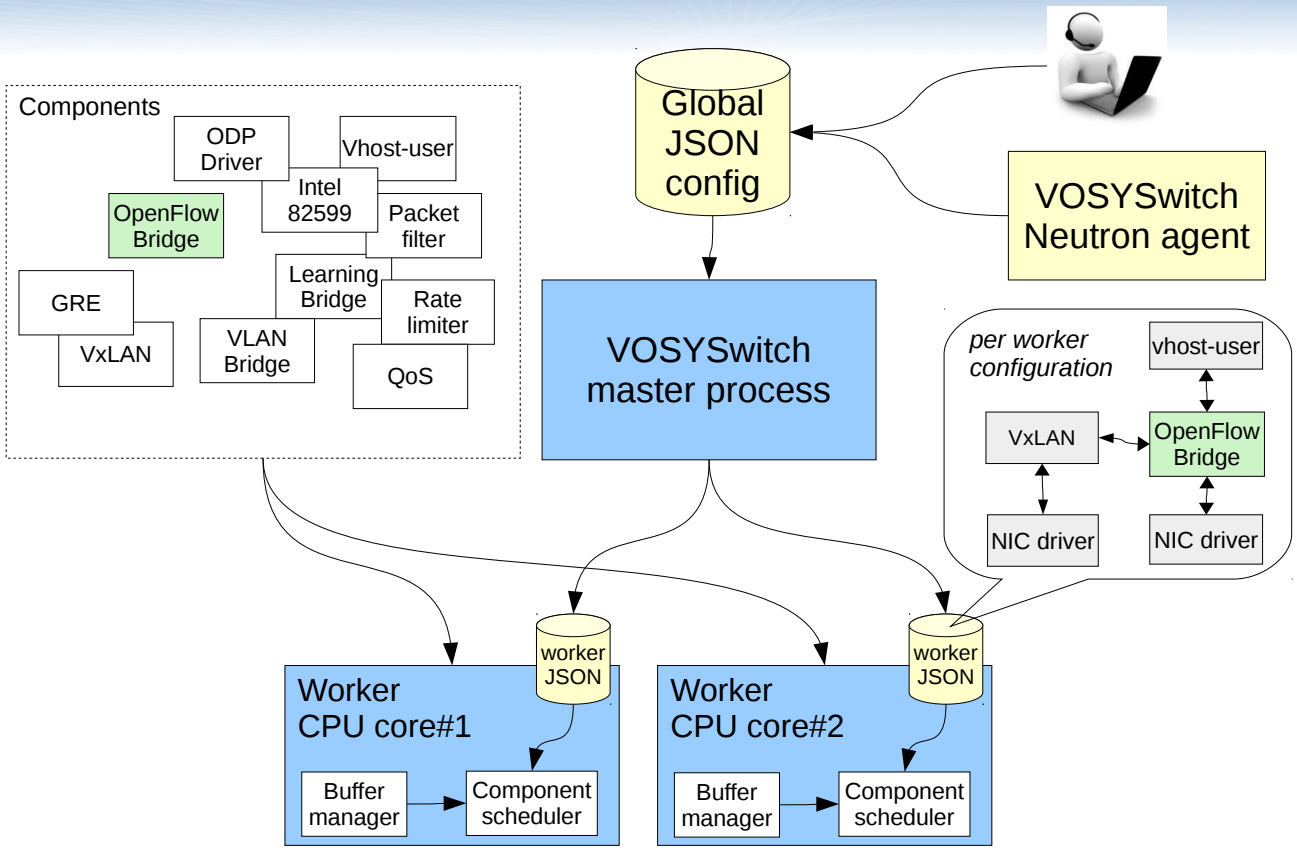
- **OpenFlow** is one of the first, and *de-facto*, standard for Software defined networking (SDN).
  - Give access to the forwarding plane of a network switch or router over the network
- Benefits provided by OpenFlow:
  - Accelerate new features and service introduction
  - Centralized control of multi-vendor environments
  - Reducing complexity of the management
  - Decoupling hardware and software, Control plane and forwarding, physical and logical configuration.

VOSYSwitch Architecture

- Based on open-source project **Snabb**
- Network is design as a **packet processing graph**
- Written using the *Lua* language
- Take advantage of the *LuaJIT* trace compiler
  - Allow self-adaptable runtime code generation
  - Dynamic code path optimization
- **User-space polling mode** drivers
- Processing graph is described in a *JSON* configuration file



VOSYSwitch Architecture



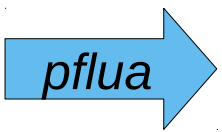
PCAP filter language

- Powerful and complete Packet filtering language
- Developed for the popular tool *tcpdump* (also used in *Wireshark*)

```
tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)
```

- Alternative library, ***pflua*** is able to compile PF filters into Lua Language:

```
ether proto 0x800 and proto 6
```




```
local cast = require("ffi").cast
return function(self,P,length)
   if length < 34 then goto L5 end
   do
      if cast("uint16_t*", P+12)[0] ~= 8 then goto L5 end
      if P[23] == 6 then return true end
      goto L5
   end
::L5::
   return false
end
```

PFmatch

- Pflua also implements a variant languge named ***pfmatch***
- Allow to associate an action to each filter

```
match {
  ((ether proto 0x800) and (proto 6) )=> ins1()
  => ins2()
}
```

pflua

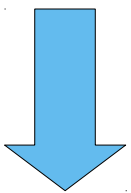
```
local cast = require("ffi").cast
return function(self,P,length,user)
  if length < 34 then goto L5 end
  do
     if cast("uint16_t*", P+12)[0] ~= 8 then goto L5 end
     if P[23] == 6 then return self:ins1(P, length, user) end
     goto L5
  end
::L5::
  return self:ins2(P, length, user)
end
```

Fit well datapath standard such as OpenFlow!

Pflua-based OpenFlow implementation

Applied to OpenFlow ...

```
(1)
     OXMXMT_OFB_IP_PROTO = 6
     OFPXMT_OFB_ETH_TYPE = 0x800
(2)
     -
```



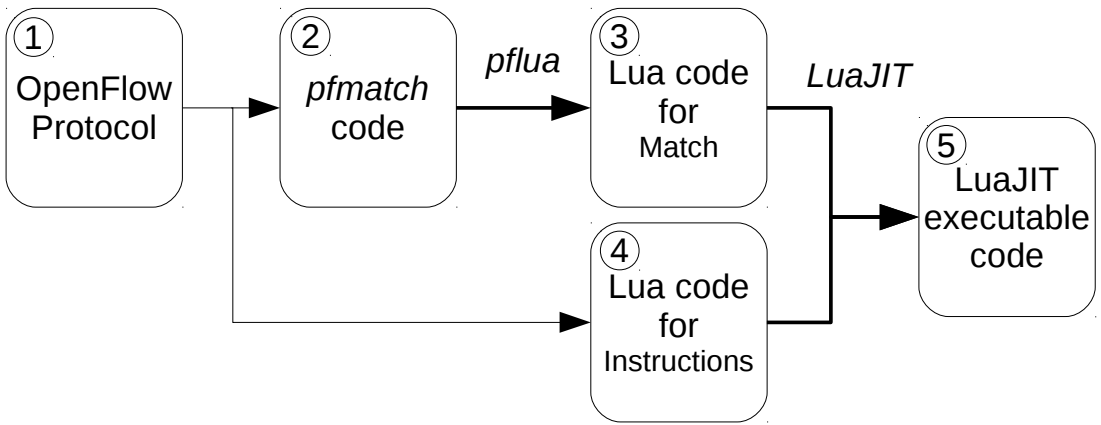
```
match {
  ((ether proto 0x800) and (proto 6) )=> ins1() -- (1)
  => ins2()                                     -- (2)
}
```

pflua

```
local cast = require("ffi").cast
return function(self,P,length,user)
  if length < 34 then goto L5 end
  do
     if cast("uint16_t*", P+12)[0] ~= 8 then goto L5 end
     if P[23] == 6 then return self:ins1(P, length, user) end
     goto L5
  end
::L5::
  return self:ins2(P, length, user)
end
```

OpenFlow implementation

- Implemented as a bridge component (usable in the processing graph)
- Translate OpenFlow flows dynamically into Lua via the PCAP filter language
- The code is generated on demand and tailored to the flow entries of the tables

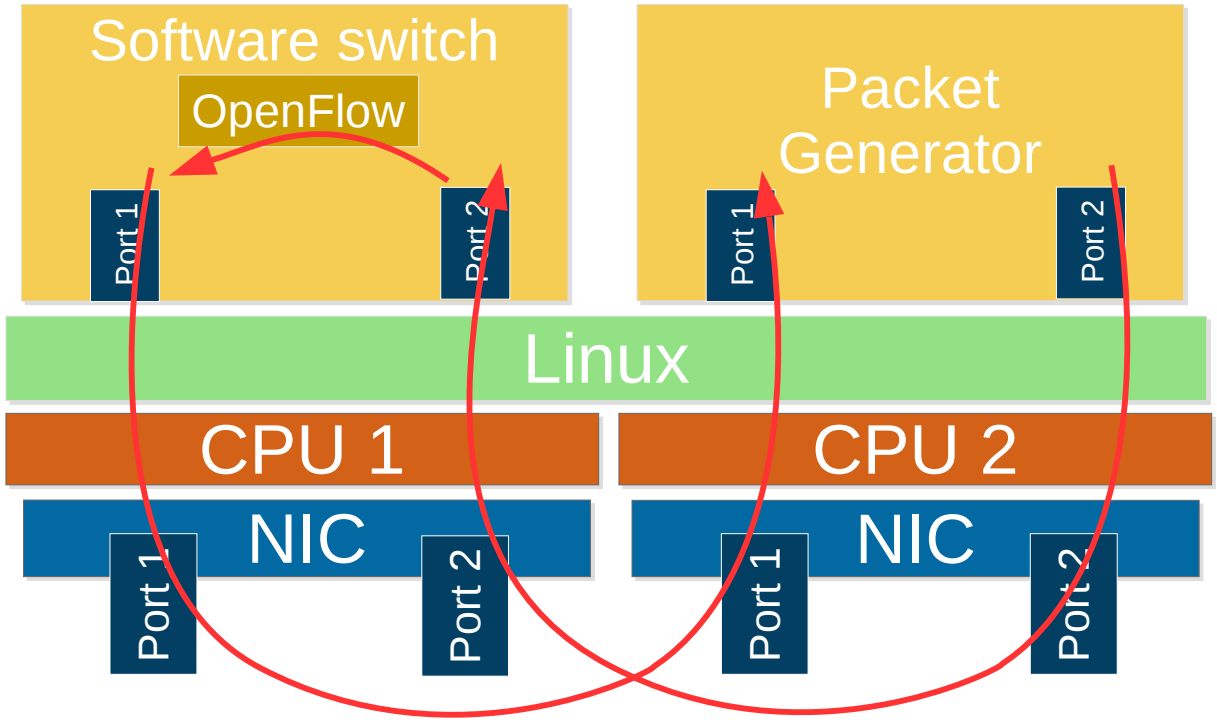


Benchmark configuration

- Comparison between **OVS-DPDK** and **VOSYSwitch**
- Same machine (Intel Xeon E5-2697 v3 @2.6GHz)
- Intel X520, 10G dual port NIC (Intel 82599ES)
- Software packet generator

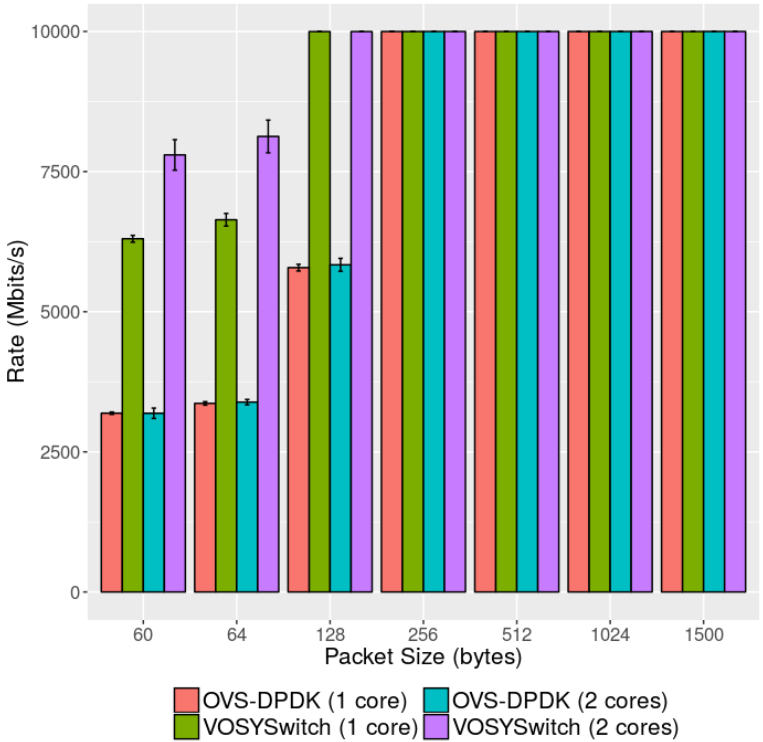
- Test scenarios with L2 and L3 flow rule types

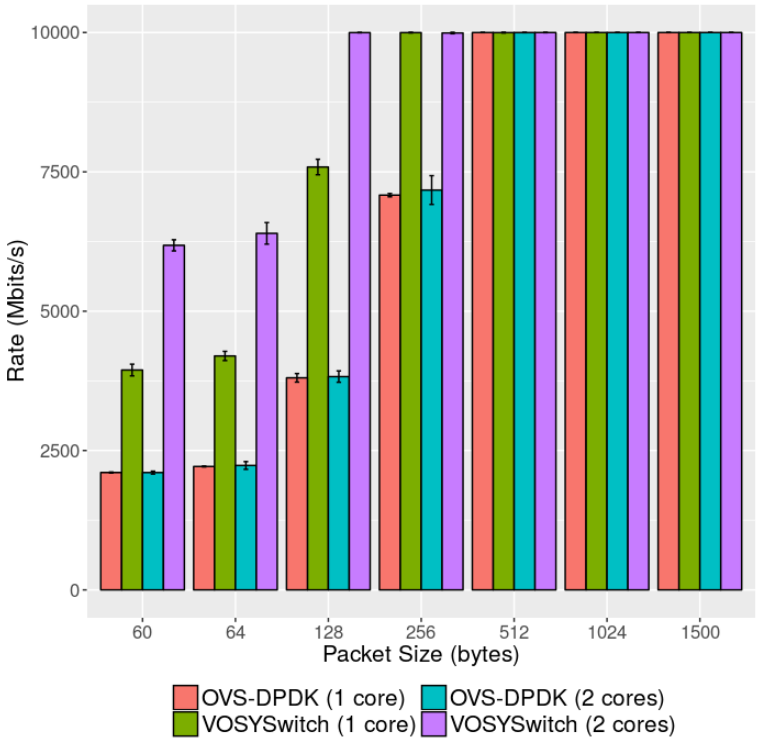
Benchmark configuration



Benchmark results

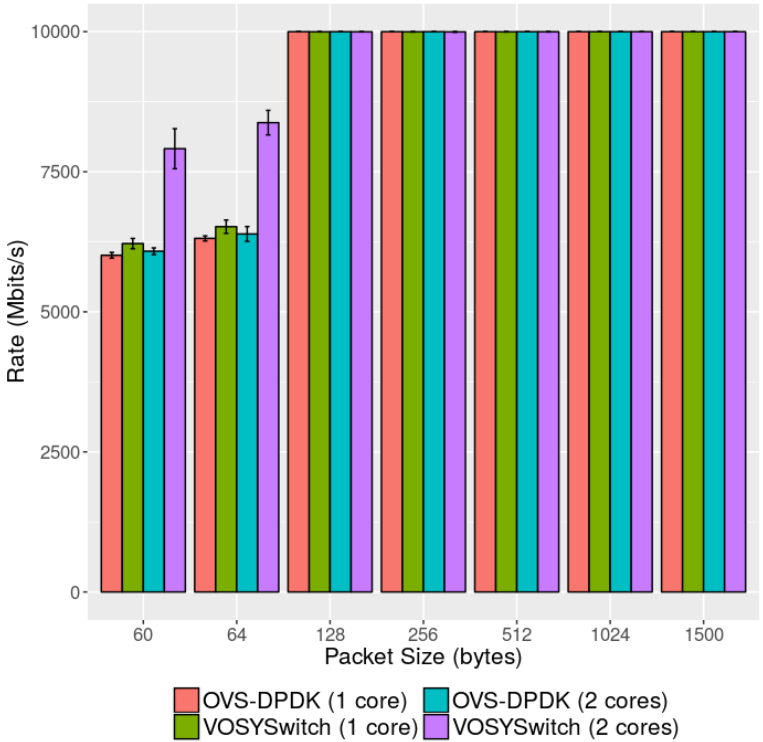
Forwarding rate with 10 and 100 flow entries matching on L2 header

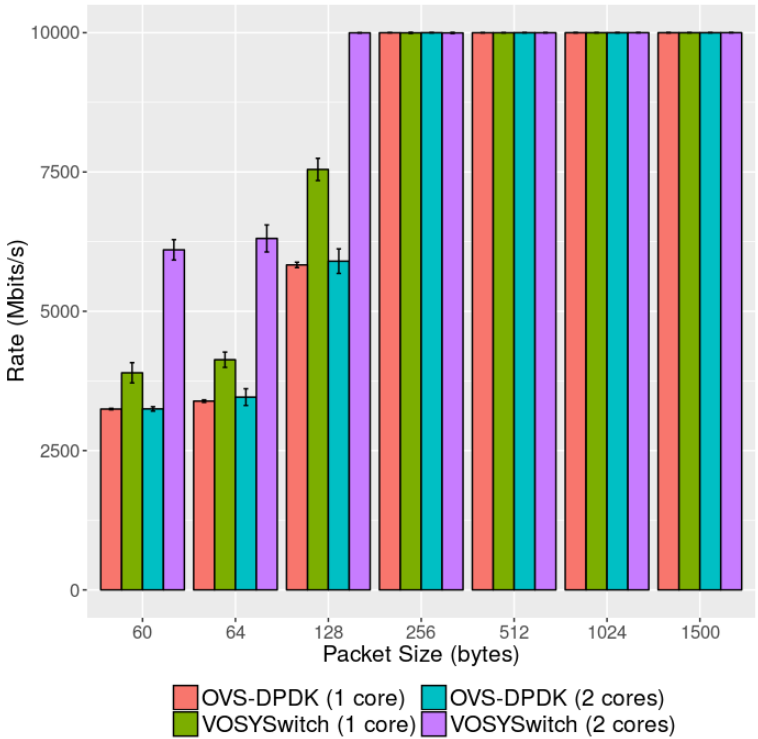




Benchmark results

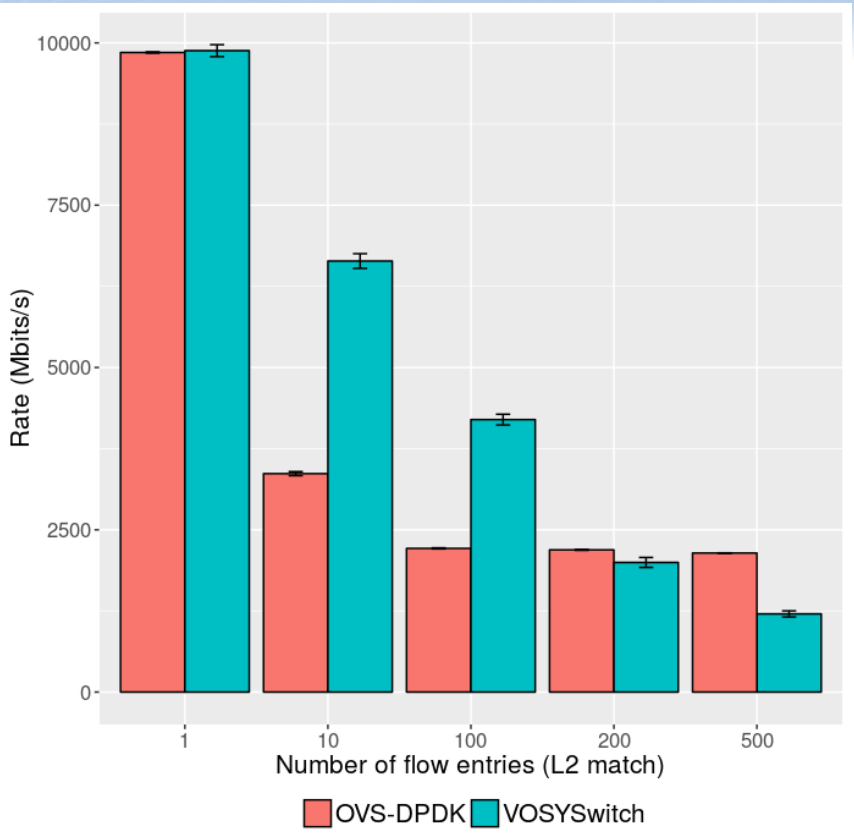
Forwarding rate with 10 and 100 flow entries matching on L3 header





Benchmark results

Forwarding rate of 64B packet
vs. number of L2 flows



Conclusion

- PCAP filter can be used to implement a full-feature dataplane
- Performance can compete with first-class software switch such as OVS-DPDK

- Future Work:
  - Cache mechanism
  - More extensive benchmarks

Thank you

Virtual Open Systems