Virtual Open Systems

VOSYSwitch port to ARMv8

and ODP integration

Nikolay Nikolaev
Alexander Spyridakis

Agenda

- **Snabb and VOSYSwitch**
- Porting to ARMv8
- DEMO description and performance results
- Further development

Introducing VOSYSwitch

Virtual Open Systems developed a vSwitch solution based on the open source Snabb network toolkit. It is written in a high-level scripting language – Lua.

The objective was to create a pure user-space, portable solution suitable for NFV deployments.

One of the achievements that were created within this project is “vhost-user” which is now a de-facto standard for provisioning virtio-net based KVM virtual machines.

Almost all of the code is written in Lua, with some syscalls in C (now in process of migration to Lua). There is a small amount of Assembly too.

Packet processing in a high-level language

Lua (/ˈluːə/ LOO-ə, from Portuguese: lua ['lu.(w)ɒ] meaning moon) is a lightweight multi-paradigm programming language designed primarily for embedded systems and clients. Lua is cross-platform since it is written in ANSI C, and has a relatively simple C API.

https://en.wikipedia.org/wiki/Lua_(programming_language)

The Snabb toolkit implements a high speed, user space, packet processing in Lua. The secret sauce to make it “fast” is the JIT compiler.

Packet processing on a high-level language - continued

LuaJIT is a multi-architecture (x86, PPC, ARM, MIPS), multi-platform (Linux, BSD, OSX, Windows) tracing JIT compiler.

Tracing just-in-time compilation is a technique used by virtual machines to optimize the execution of a program at runtime. This is done by recording a linear sequence of frequently executed operations, compiling them to native machine code and executing them. This is opposed to traditional just-in-time (JIT) compilers that work on a per-method basis..

https://en.wikipedia.org/wiki/Tracing_just-in-time_compilation

Snabb – the upstream project

As most of the software stacks, snabb is composed of several building blocks

- Snabb core – where the “main” lives
- Library – tools and utilities
- Apps – the building blocks
- Programs – a set of apps chained together

Snabb – the upstream project continued

The available Snabb apps are:

- Intel 82599 10Gbps driver with VMDq support; Solarflare
- Vhost-user and virtio-net on the device side
- Virtio-net driver, for running it in VMs
- Packet filter with connection tracking; Rate limiter
- IP/GRE tunnels
- Learning bridge
- Single process/ single thread
- Linux/x86_64 only

Snabb – a simple program

A simple example program to connect a Intel NIC to a VM with virito

```
local Intel82599 = require("apps.intel.intel_app").Intel82599
local VhostUser = require("apps.vhost.vhost_user").VhostUser

local c = config.new()
config.app(c, "nic", Intel82599, {pciaddr = "0000:00:01.00"})
config.app(c, "vh1", VhostUser, {socket_path="/tmp/vh1.sock"})

config.link(c, "nic.tx -> vh1.rx")
config.link(c, "vh1.tx -> nic.rx")

engine.configure(c)
engine.main()
```

VOSYSwitch – a vSwitch based on Snabb

Virtual Open Systems created VOSYSwitch solution for the end user, by amending Snabb features with:

- JSON configuration to define a packet processing graph
- Multi-process, single thread
- OpenStack Mitaka integration
- Performance optimizations (VM2VM)
- Packet scheduler – PQ, FQ, WFQ
- A software switch with VLAN and IGMP support
- RPM/DEB packaging, systemd integration

VOSYSwitch – work in progress

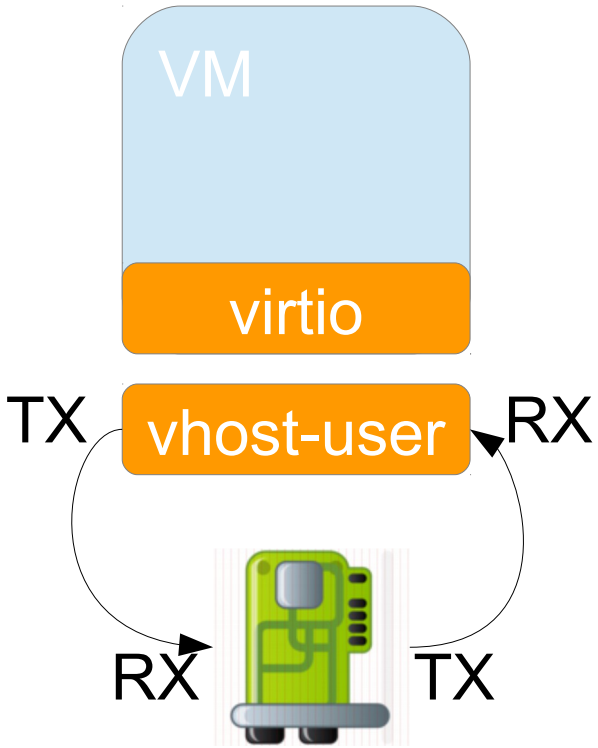
Virtual Open Systems is currently working on a number of new (substantial) features which are to be released this year:

- OpenFlow 1.4 support
- More overlay networks - VxLAN, NSH
- ARM/v8
- ODP

VOSYSwitch – JSON configuration example

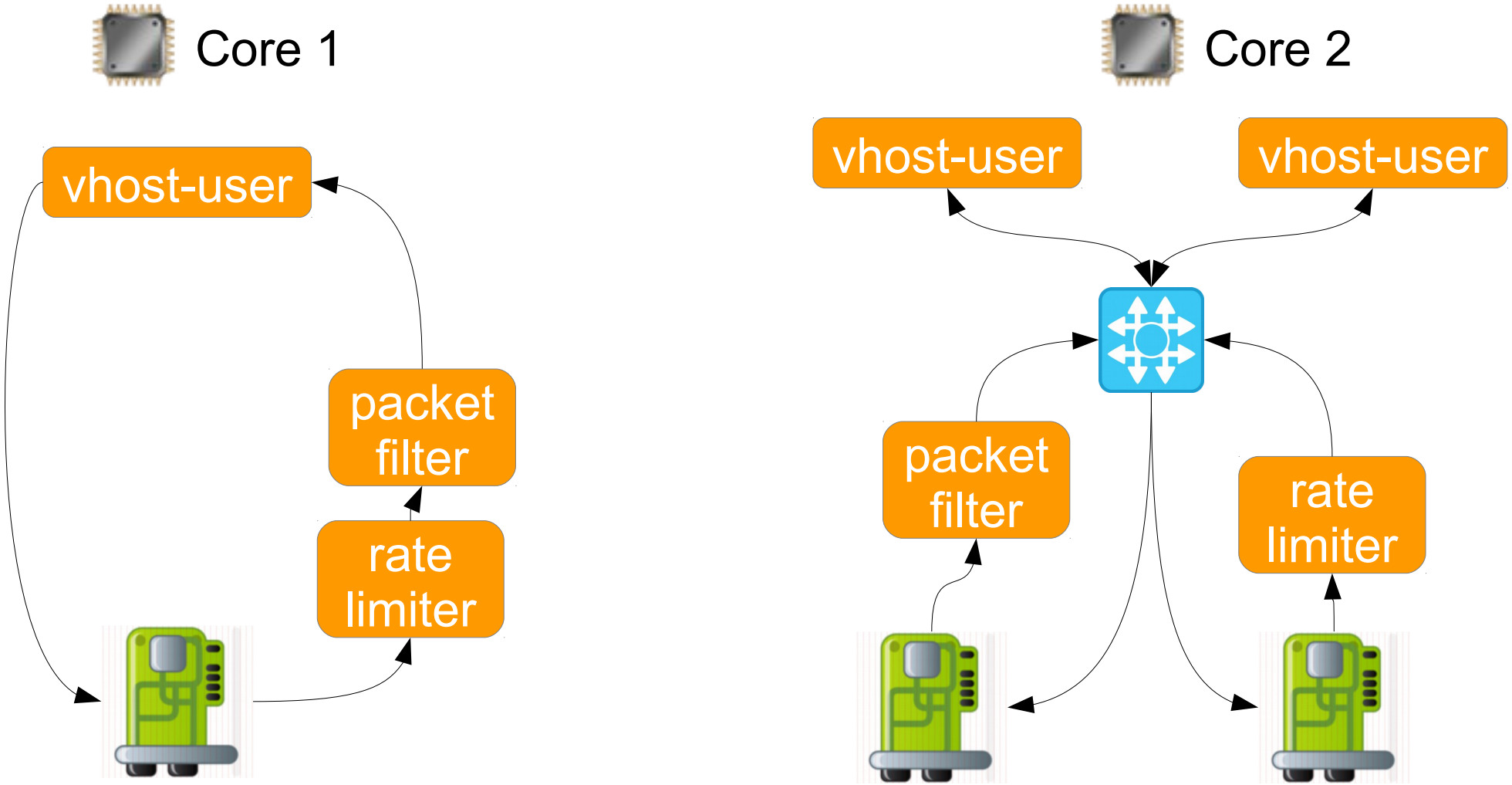
The previous example implemented in VOSYSwitch's JSON:

```
{
    "switch1" : {
        "core" : "0x1",
        "devices" : {
        "igb1" : {
                "type" : "EthPCI",
                "args" : {"pciaddr" : "0000:XX:00.1"},
                "links" : { "tx": "vh1.rx"}
            },
        "vh1" : {
                "type" : "VhostUser",
                "args" : {"socket_path" : "/tmp/vh1.sock"},
                "links" : { "tx": "igb1.rx"}
            }
        }
    }
}
```



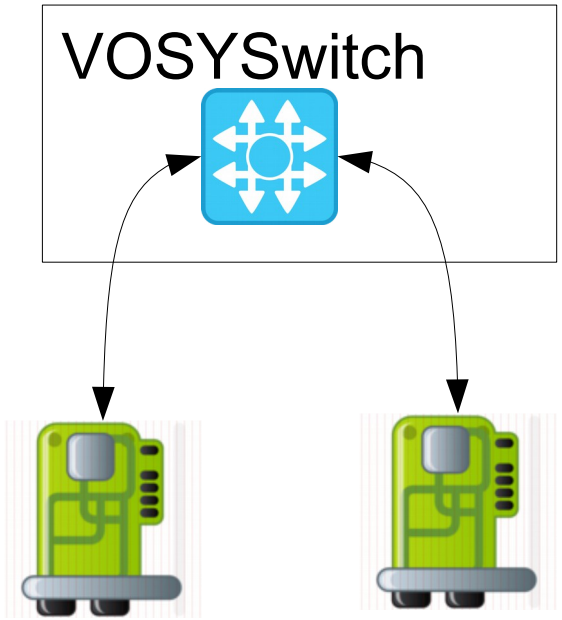
VOSYSwitch – more examples

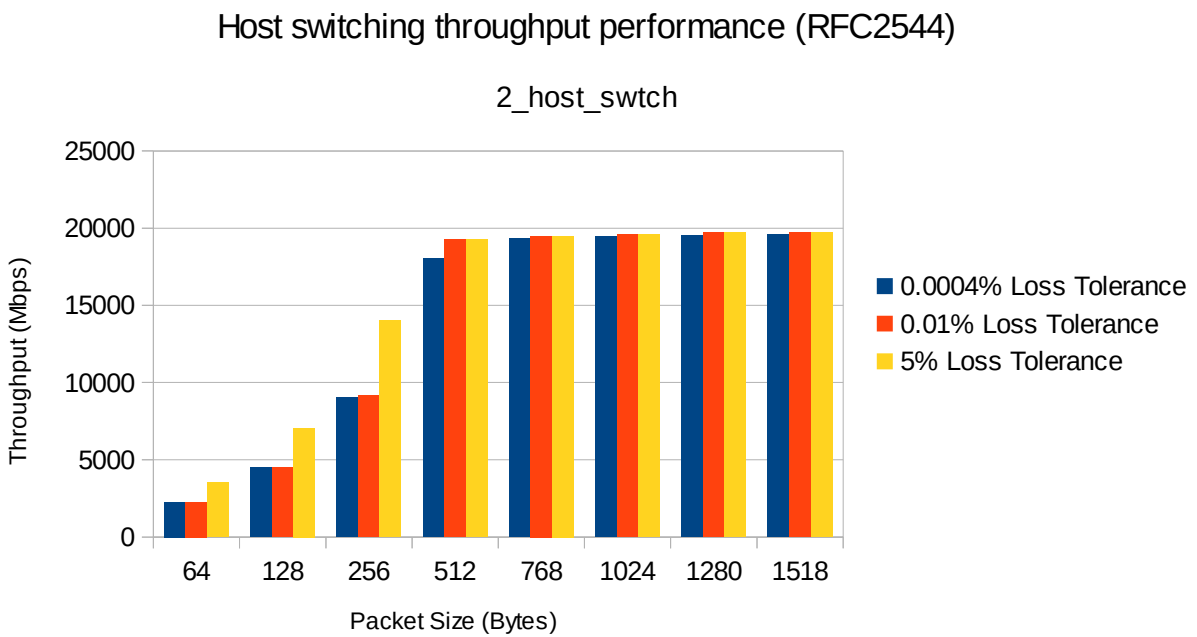
More complex topologies within the same configuration:



VOSYSwitch – performance on x86_64

Ixia RFC2544, simple vSwitch with 2 interfaces on the host





Agenda

- Snabb and VOSYSwitch
- **Porting to ARMv8**
- DEMO description and performance results
- Further development

VOSYSwitch port to ARMv8 - challenges

The porting effort started with a lot of low-level details:

- LuaJIT limitations:
  - AArch64 - interpeter
  - AArch32 – ARMv7 full support
  - 47bit VA limitation (LuaJIT/issues/49)
- Snabb limitations (VA/PA conversion by mask)
- 64k vs 4k pages
  - AArch32 compat mode
- SSE/AVX port to NEON
- gcc-armhf -march=armv8-a+crc

VOSYSwitch – ODP adoption

The initial effort was done with linux-generic on x86. Mostly without problems.

- Modules used *odp_pool, odp_packet, odp_pktio*
- API inconsistencies, even in the same API level
  - `#define ODP_EVENT_INVALID _odp_cast_scalar(odp_event_t, 0xffffffff)`
  - `#define ODP_EVENT_INVALID _odp_cast_scalar(odp_event_t, NULL)//in ODP-DPK`
- Inline functions in the API headers
- platform/linux-dpdk/include/odp/packet.h

```
static inline uint32_t odp_packet_len(odp_packet_t pkt) {
    return *(uint32_t *)(void *)((char *)pkt + pkt_len_offset);
}
```

ODP-DPDK compilation on ARMv8

DPDK's ARM support is still not mature enough. The main ODP-DPDK development was done in x86. Some of the issues found:

- odp_pktio_send() implementation returns -1 when no packets have been sent which is considered an error on ODP side, but not on DPDK side.
- -msse4.2 – This code was never compiled for ARM?
- virtio-net driver for ARM
- dpdk_memcpy on arm/arm64 is a #define, so this does not work:

```
return (*dpdk_memcpy)(dst, src, num);
```

Agenda

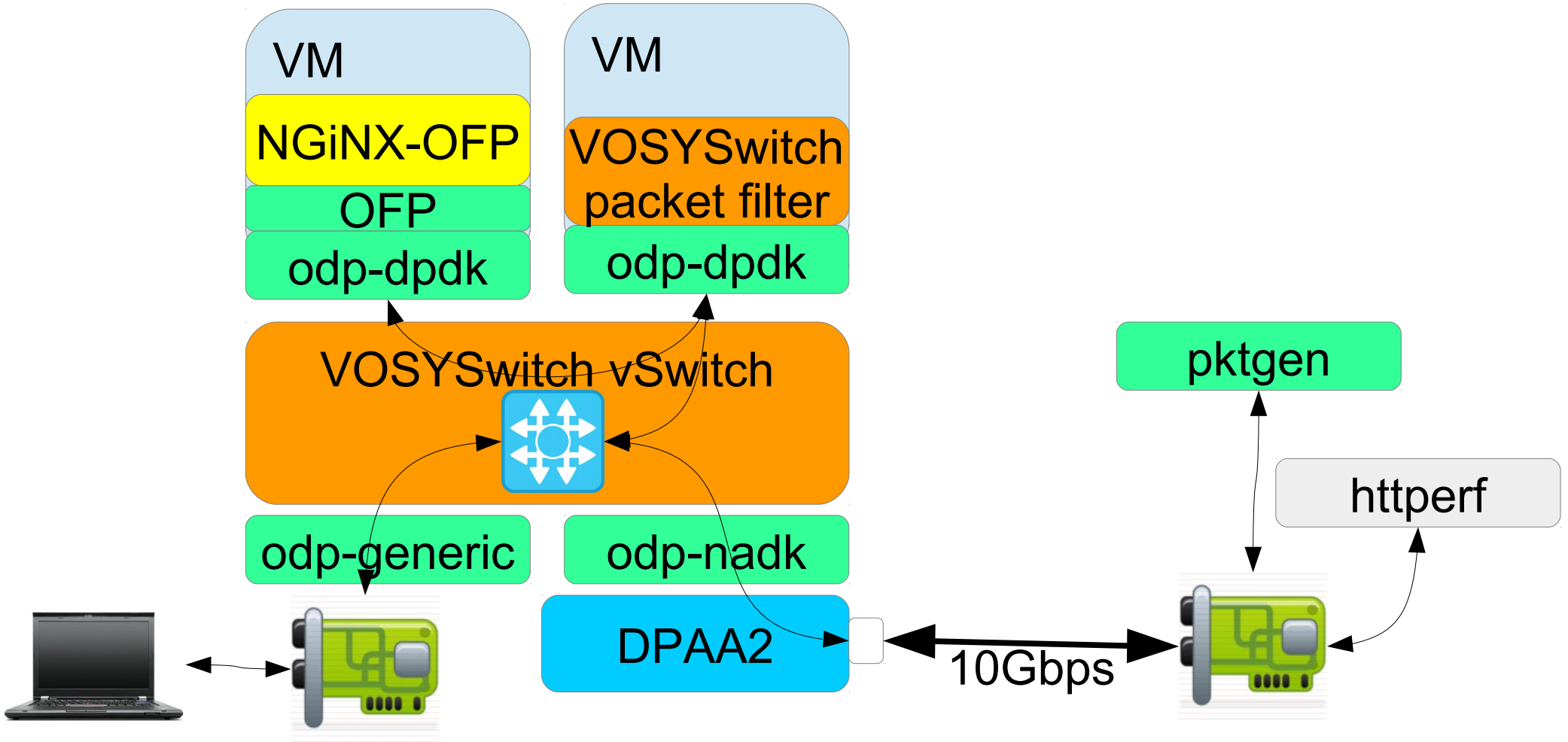
- Snabb and VOSYSwitch
- Porting to ARMv8
- **DEMO description and performance results**
- Further development

VOSYSwitch – ODP integration

By Leveraging the ODP API, VOSYSwitch achieved very important goals.

- Extend the number of supported platforms:
  - ODP-linux-generic
  - ODP-DPDK
  - ODP-NADK
- Single binary running on multiple hosts
  - VOSYSwitch for ARM running on both LS2085A and Juno board
- Single binary can access multiple HW resources at once, i.e. DPAA2 + Intel DPDK

VOSYSwitch running on LS2085ARDB



VOSYSwitch – vSwitch JSON

```
{
    "switch1" : {
        "core" : "0",
        "devices" : {
...
        }
    }
    "switch2" : {
        "core" : "1",
        "devices" : {
            "vh_fw_out" : {
                "type" : "VhostUser",
                "args" : {"socket_path" : "/tmp/vh_fw_out.sock", "is_server" : true},
                "links" : { "tx": "odp0.rx"}
            },
            "odp0" : {
                "type" : "EthODP",
                "args" : {"ifname" : "dpni-0", "platform" : "nadk"},
                "links" : { "tx": "vh_fw_out.rx"}
            }
        }
    }
}
```

VOSYSwitch – packet filter JSON

```
{
    "switch1" : {
        "core" : "0x1",
        "devices" : {
            "odp_in" : {
                "type" : "EthODP",
                "args" : {"ifname" : "0", "platform" : "dpdk", "platform_params" : "-m32"},
                "links" : { "tx": "odp_out.rx"}
            },
            "fw" : {
                "type" : "Firewall",
                "args" : {"filter" : "arp or icmp or tcp port 80", "state_table" : true},
                "links" : {"tx" : "odp_in.rx"}
            },
            "odp_out" : {
                "type" : "EthODP",
                "args" : {"ifname" : "1", "platform" : "dpdk", "platform_params" : "-m32"},
                "links" : { "tx": "fw.rx"}
            }
        }
    }
}
```

DEMO performance results

Using the Apache Bench tool to get the number of connections per second which the demo infrastructure can handle. Using 10 parallel streams we are able to handle 5000 connections, using single two dual core VNFs and a single core vSwitch.

- `ab -n 10000 -c 1 100.0.0.4/`
  - 1500 conn/s
- `ab -n 10000 -c 10 100.0.0.4/`
  - 5000 conn/s

Agenda

- Snabb and VOSYSwitch
- Porting to ARMv8
- DEMO description and performance results
- **Further development**

VOSYSwitch – next steps for ARMv8 and ODP

ODP API proved to be very flexible and more parts of it will be adopted.

- ODP_CRYPTO
- ODP_QUEUE, ODP_SCHEDULER
- ODP_CLASSIFICATION

Virtual Open Systems is already investigating into a better LuaJIT support for ARMv8. It is becoming a major supported platform in VOSYSwitch and bringing the best performance and experience out of it is a major goal.

VOSYSwitch Roadmap

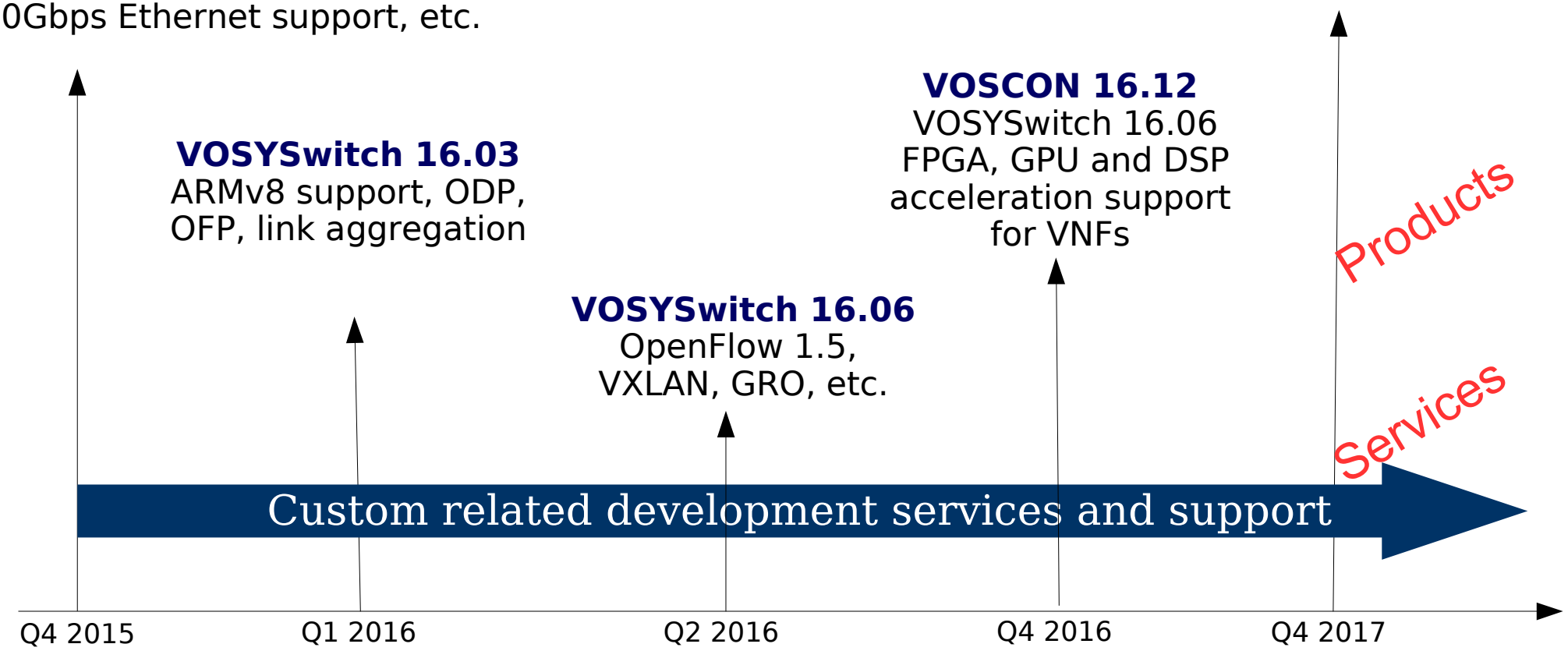
VOSYSwich 15.12
VMtoVM, IPv6 GRE tunnels, filtering, traffic limiter, OpenStack Mitaka support, QoS, VLAN and IGMP, 40Gbps Ethernet support, etc.

VOSCON 17.06
Multitenancy support
vTPM and RT, HA

VOSCON 16.12
VOSYSwitch 16.06
FPGA, GPU and DSP acceleration support for VNFs

VOSYSwitch 16.03
ARMv8 support, ODP, OFP, link aggregation

VOSYSwitch 16.06
OpenFlow 1.5, VXLAN, GRO, etc.



ODP future suggestions

Could we get some of these in the future:

- Tunneling interfaces offload – VxLAN, GRE
- OpenFlow switch

Questions

See our demo

In the LNG room

and on

www.virtualopensystems.com

n.nikolaev@virtualopensystems.com

a.spyridakis@virtualopensystems.com

Virtual Open Systems