The Evolution of Open vSwitch

LinuxCon 2014

Jesse Gross

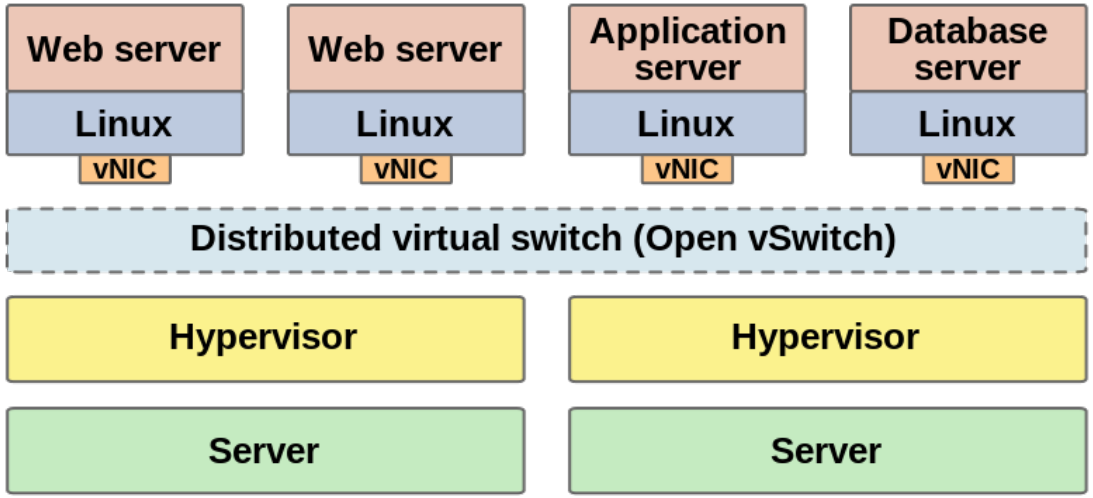
August 21, 2014

vmware®

A Very Brief Introduction to OVS

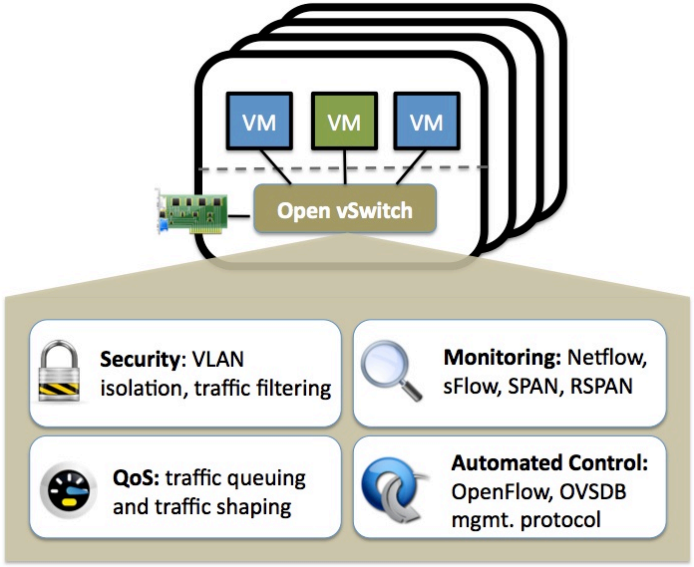
A programmable switch!

- Optimized around large scale automation and control
- Efficiently handle large amounts of state
- Distribute state around the network



Where We Started (in ~2009)

- Truly began life (in 2007) as an academic project
- Original uses were for enforcing policy and security
- Later turned into what is now OpenFlow
- Reborn as an open source project in 2009

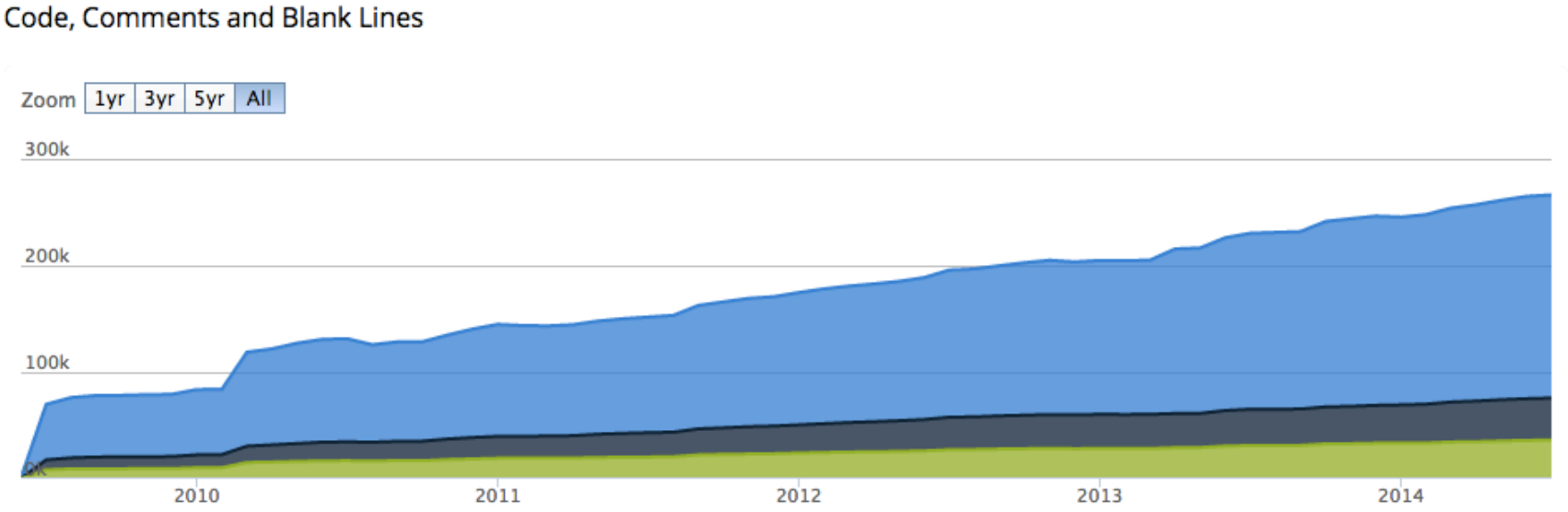


Where We Are Today

- Used in most SDN/network virtualization systems
- Reference implementation for OpenFlow
- Large feature set beyond OpenFlow
- Integrated with Linux, libvirt, cloud management tools
- Ported onto several switching ASICs
- Incorporated into many vendors' products (code is Apache licensed)

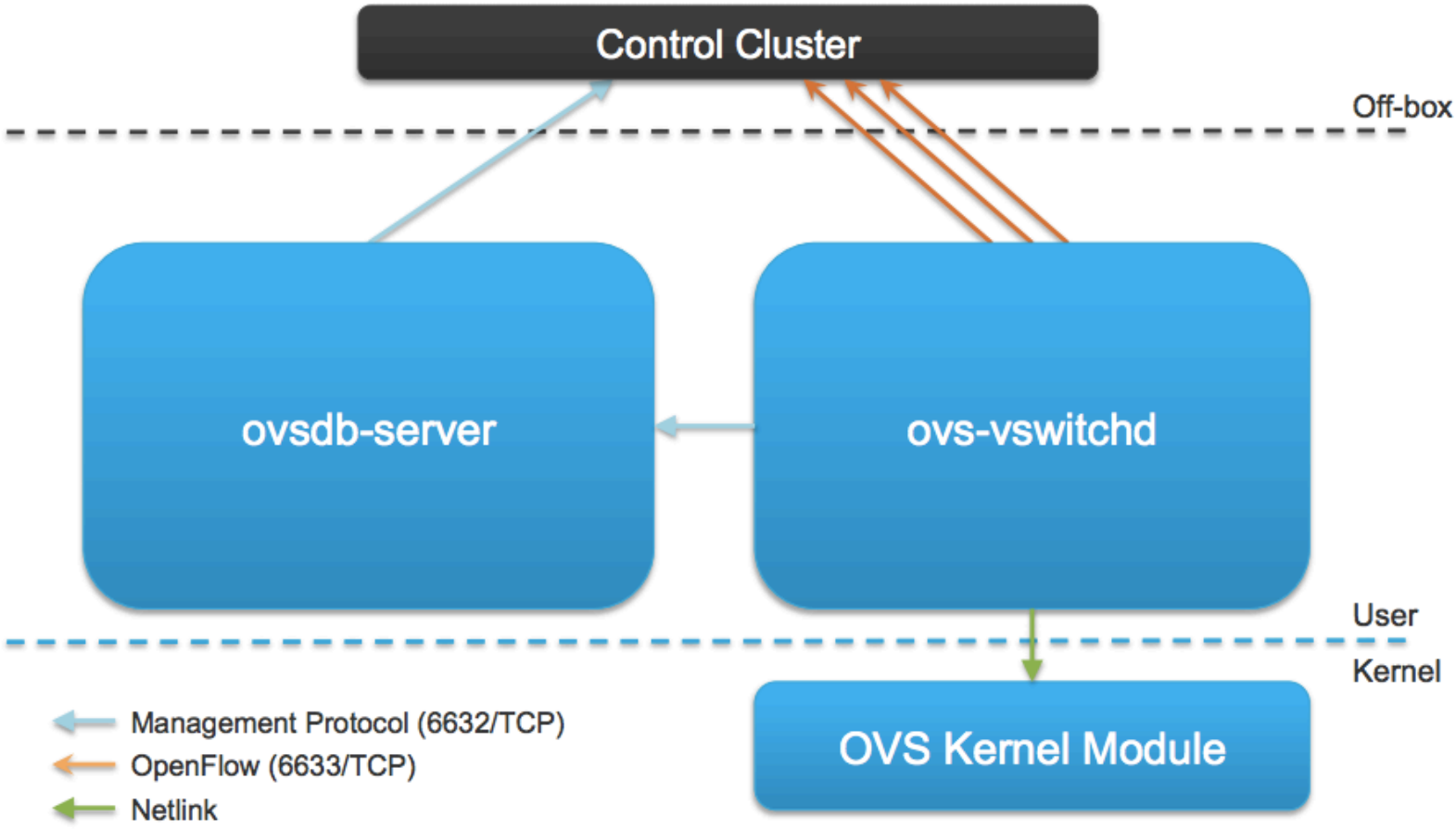
Used by the majority of OpenStack deployments as the network switch

In Between



vmware

Open vSwitch Architecture



Challenges

“No plan survives contact with the enemy.”

Issues in Production Deployments

- Performance
- Performance
- Performance

Why?

Not actually that surprising:

- Open vSwitch design goal is flexibility and programmability
  - ➔ Usually this isn't free
- Common use cases usually involve large amounts of state
- Software sometimes has a bad name in networking performance
  - ➔ If you are using virtualization, you are probably already using a software switch.

What Does ‘Performance’ Mean?

Many ways to measure network performance:

- Bandwidth (bytes per second)
- Small packets (packets per second)
- Flow setup rate
- Latency
- CPU usage

Throughput

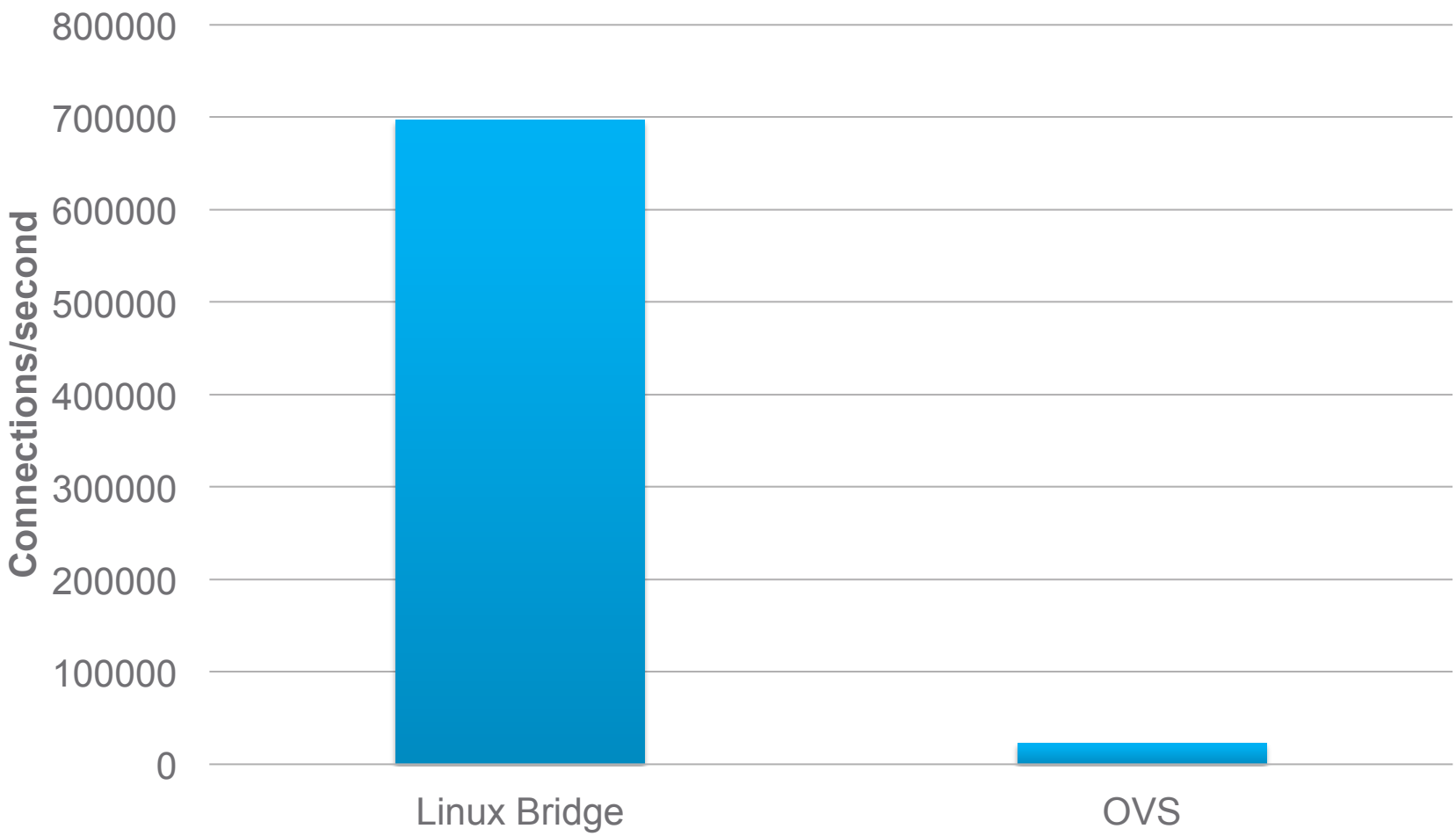
Common configuration: 10G NIC, bulk traffic

9.88 Gbps

(i.e. full rate after packet headers)

Dual socket Sandy Bridge class CPUs, Spirent traffic generator

New Connections



Megaflows

OVS maintains a set of flow entries in the kernel for established traffic

Good:

- Hits provide good performance even for complex rules

Bad:

- Missing the entry

Idea: Dynamically change the definition of a flow based on rules.

L2 learning switch:

Before: Input port, Ethernet src/dst, IP src/dst, TCP src/dst, …

After: Input port, Ethernet src/dst

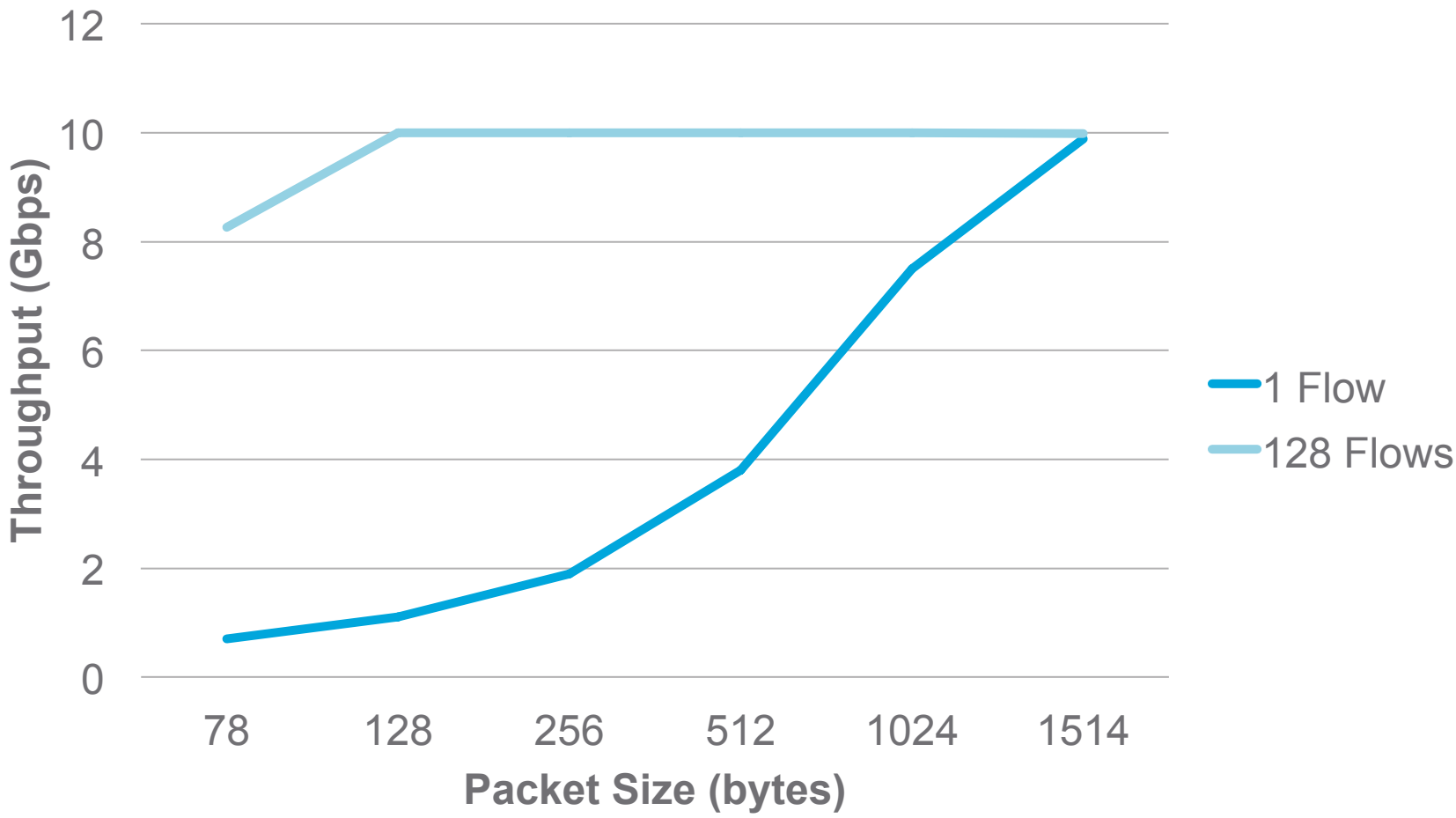
Megaflows: Results

Megaflows largely either work or they don't:

- If wildcards match, performance is the same as established flows.
- If no match, performance is the same as without megaflows.

Wildcard set (and performance) depends on OVS configuration.

Small Packets



Dataplane Development Kit (DPDK)

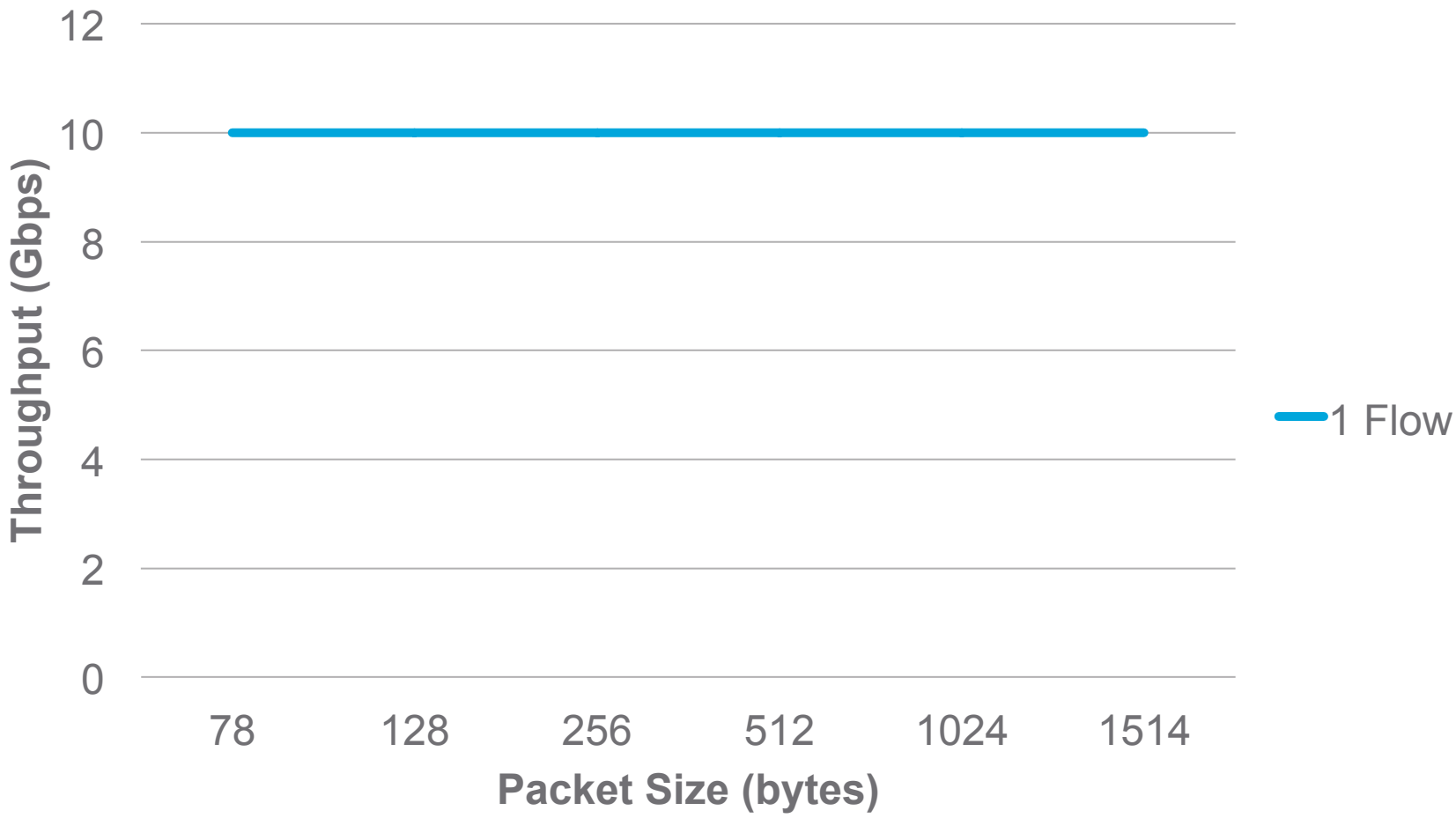
What is DPDK?

- A packet processing environment in userspace
- Application specific, so no general purpose overhead
- Set of libraries to take advantage of modern processor features
- Run-to-completion model, takes over one or more cores fully

OVS kernel code is fairly small and can be ported to DPDK as if it was a different operating system.

DPDK is almost like a hardware offload – but on x86 cores.

DPDK: Results



Hardware Offload

Large variety of hardware offloads seen over the years

Some successful:

- Checksum offload
- Scatter/gather
- TCP segmentation offload (TSO)
- Receive hashing/spreading

Some not:

- TCP offload engine (TOE)
- NIC passthrough (SR-IOV)

Level of software control is the difference between the categories.

Offloading OVS

Some general offloads are particularly useful in OVS environments:

- Stateless offloads for tunnels
- Quality of service (QoS)
- Encryption

OVS flow table offload?

- Partial offload: NIC indicates likely match, software verifies and processes
- Full offload: Table lives in NIC

Still under discussion, no performance results yet.

Questions

(and maybe answers)