Programmable Networking with Open vSwitch

Jesse Gross

LinuxCon

September, 2013

Background: The Evolution of Data Centers

Virtualization has created data center workloads that are large, rapidly changing, and location independent.

- Enabled by a layer of software indirection between logical unit (virtual machine) and underlying hardware (physical machine).
- The virtualization layer exposes a programmable API to what previously required a human to reconfigure.
- Current networking has many of the same problems as traditional servers, limiting the benefits of virtualization.

Networking needs to be programmable

What is Open vSwitch?

Open vSwitch is an open source switching stack for virtualization.

The most powerful piece of real estate in a network is the edge and the hypervisor is the new edge.

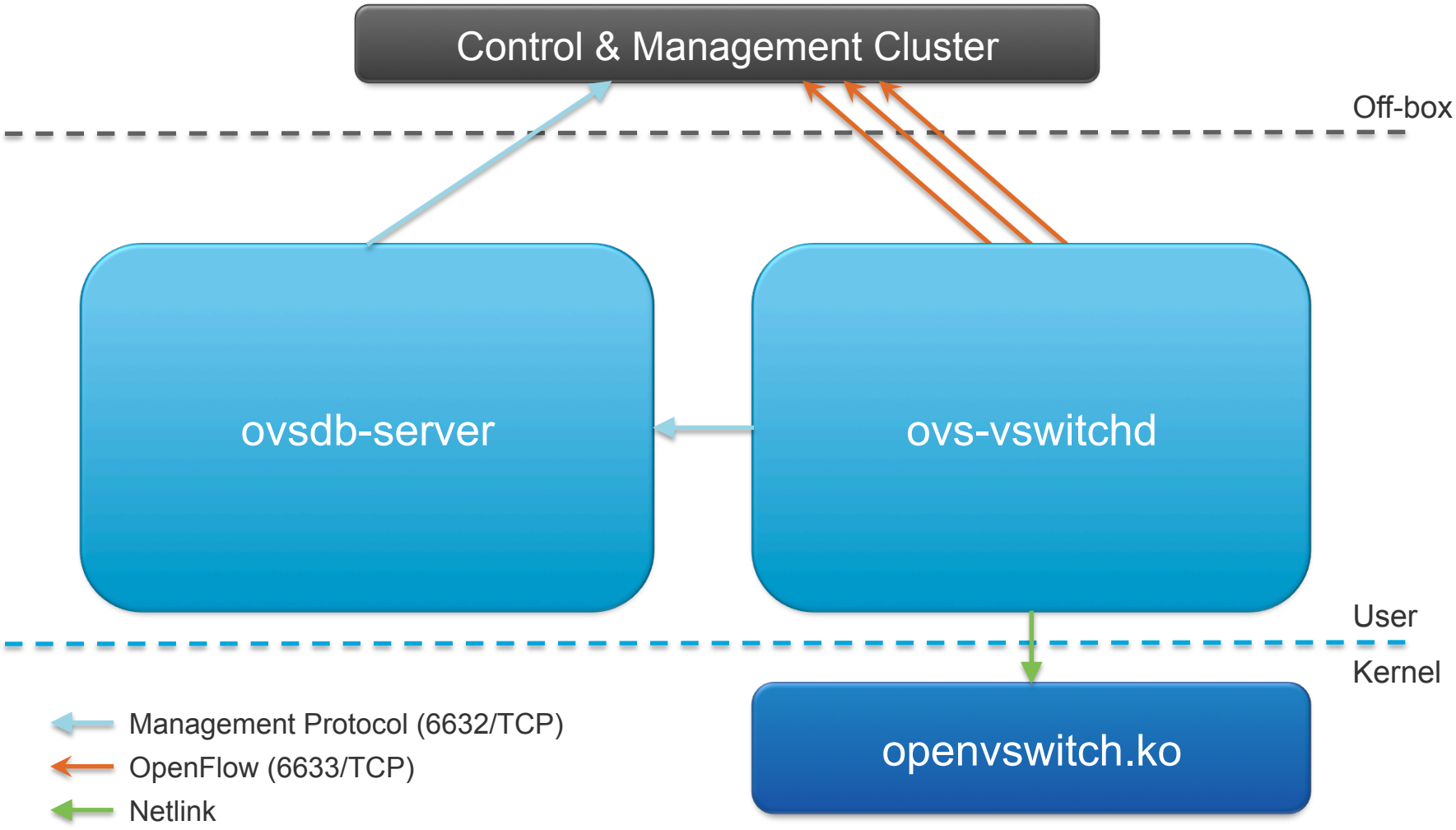
Two ways to view OVS:

- Gaining back visibility and control that usually comes from the features of a hardware switch.
- An opportunity to exploit the flexibility that comes from software and virtualization.

Open vSwitch allows you to write a program to control your network.

Sample of Contributors

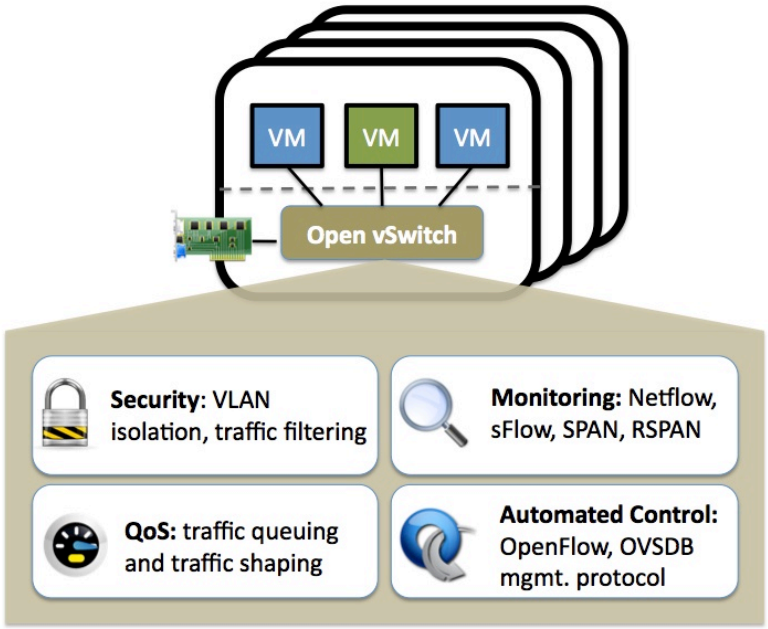
Open vSwitch Architecture



Basic Features

Open vSwitch brings many features standard in hardware devices to virtualized environments:

- VLANs
- LACP and other bonding modes
- STP
- QoS shaping and policing
- ACLs over a range of L2-L4 protocols
- NetFlow, sFlow, IPFIX, mirroring
- A variety of tunneling protocols



Plus remote programmability and management features:

- OpenFlow 1.0 and experimental support for versions 1.1-1.3.
- All features and status remotely configurable and viewable.
- Many extensions for supporting high availability control clusters.

Advanced Capabilities

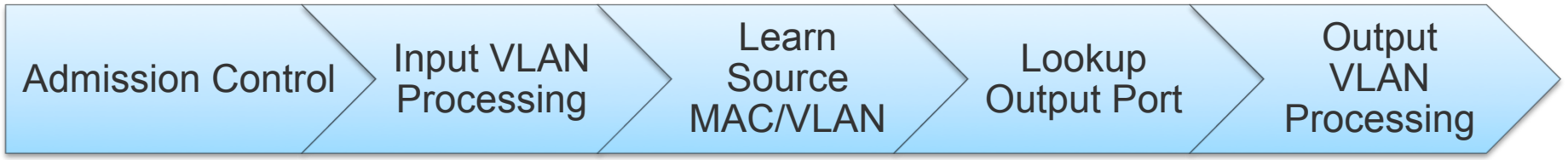
Programmability requires primitives more similar to a CPU than a network ASIC.

Over time, the flow table in Open vSwitch has slowly changed from a list of policies to a nearly general purpose processing pipeline.

Examples:

- *Resubmit:* Move between multiple independent flow tables, similar to subroutines.
- *Registers:* Storage for intermediate metadata, including manipulation functions such as a stack.
- *Learning:* Dynamically generate new flows based on packet traffic patterns.
- *Hashing and Sampling:* Perform actions based on deterministic or probabilistic properties of the traffic.

A Simple Switch Pipeline

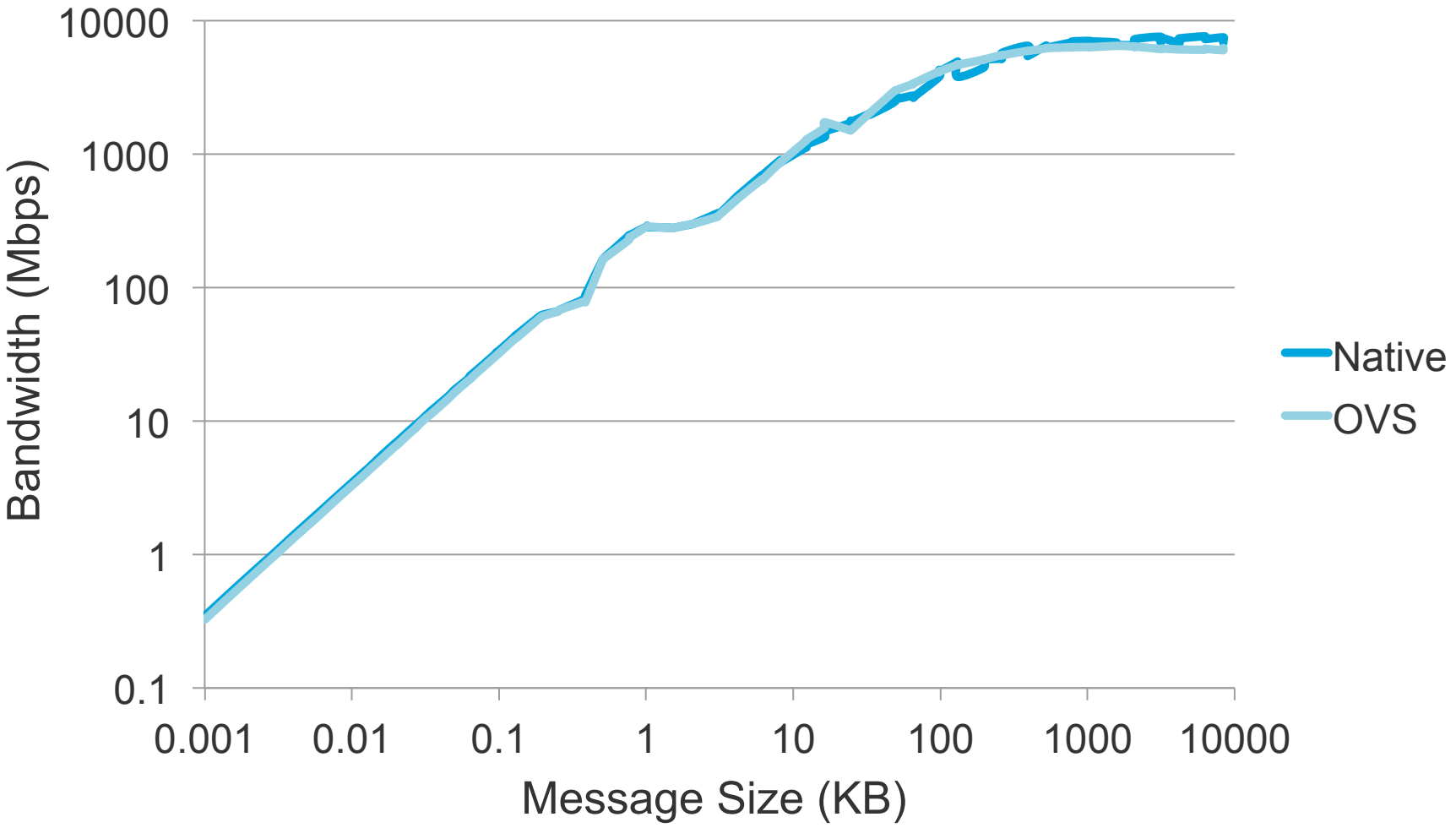


Open vSwitch makes it possible to emulate a traditional pipeline or extend for new models.

1. Flows to drop illegal packets (i.e. reserved addresses) and resubmit valid packets to the next stage.
2. Classify packets on ingress port and add VLAN tag. Resubmit to next stage.
3. Learning action to generate new flows based on source MAC, VLAN, and input port. Fields populate a template and placed in next stage. Resubmit.
4. Match flows generated by learning or use low priority flood flow. Resubmit.
5. Strip VLAN tag for access ports and output.

Performance

How does programmability impact forwarding rates?



vmware

Performance

Many aspects of performance:

Established Flows:

New flows are sent to userspace and exact match entries are installed in the kernel. All classification happens in userspace, out of the fast path. Most additional features do not affect performance.

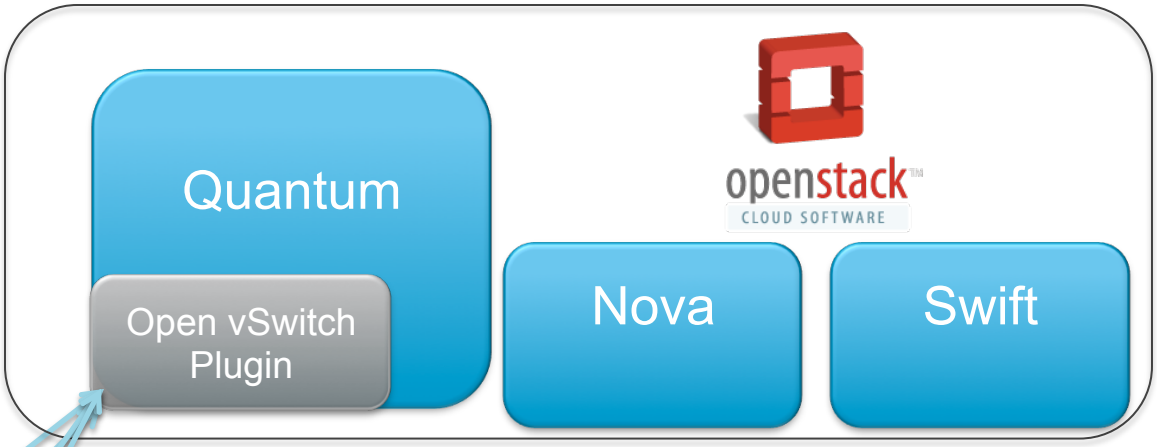
Connection Setup:

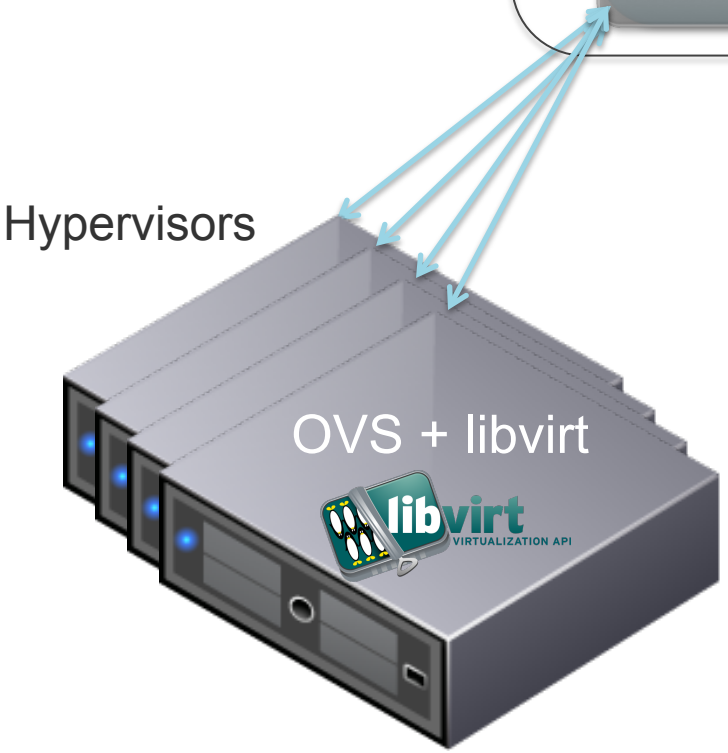
Most challenging case for Open vSwitch. Optimizations are currently under development to both reduce the number of unique flows and increase the speed of setup.

Many Sustained Connections:

A large number of flows does not directly affect throughput but the overhead of maintaining statistics increases. Heuristics are used to balance the rate of updates with overhead.

Integration Points: libvirt and OpenStack



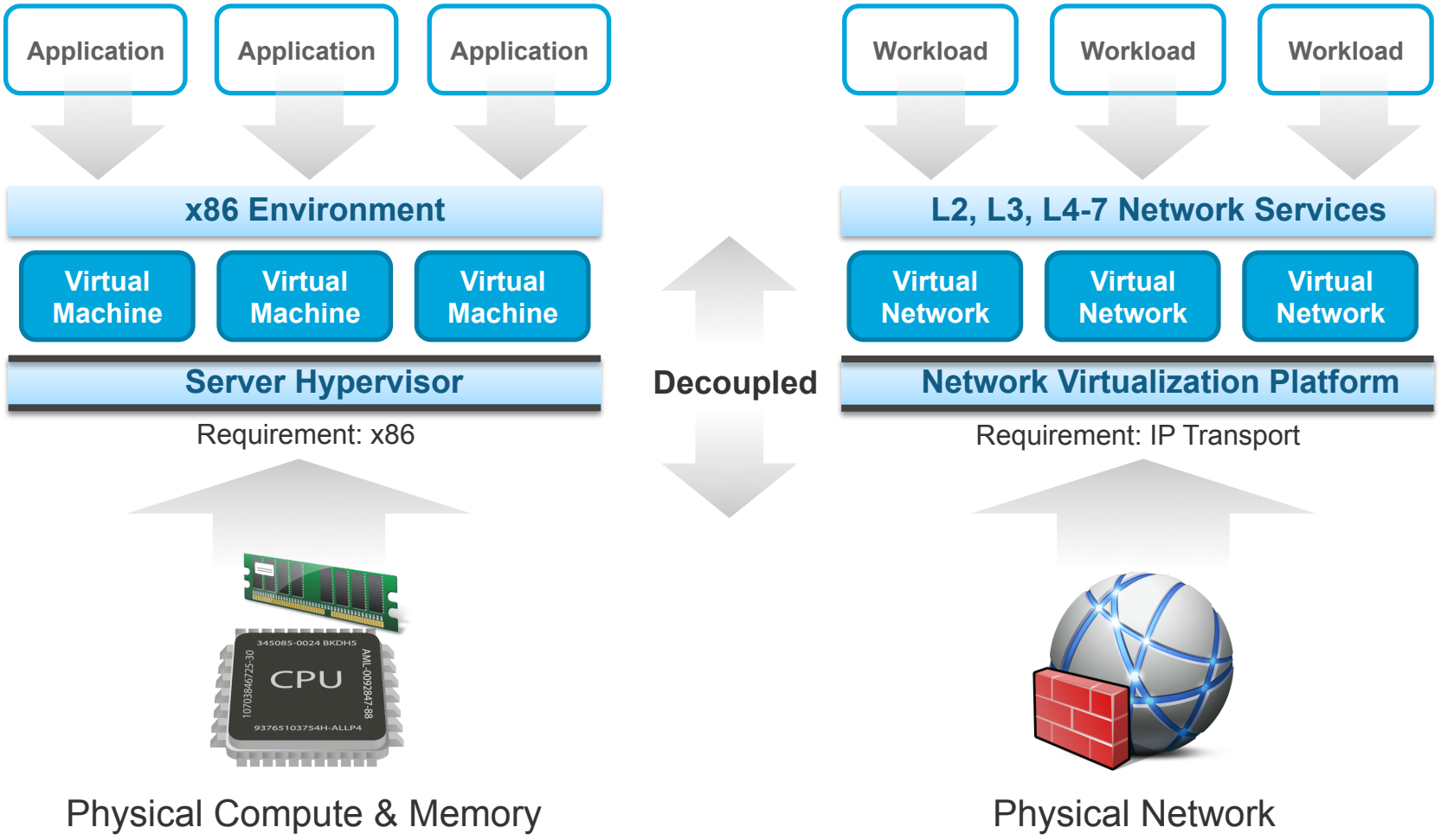


Two main touch points between Open vSwitch and the rest of the environment:

libvirt (0.9.11): VM lifecycle on hypervisors – attach ports to bridges.

OpenStack/Quantum (Folsom): Manage many Open vSwitch instances to create network abstractions.

Tying It All Together: Network Virtualization



vmware

Future Directions

Performance

- Wildcards in the kernel to reduce flow setups
- Userspace multi-threading
- General optimization

Increased Integration

- Tunnel upstreaming
- Further native support from both hypervisor and network management tools
- Additional use of Linux components, particularly for stateful features

Additional Features

- Production-ready support for OpenFlow 1.1+
- Additional protocols and networking functionality
- More programmability and controller assistance

Contribute: http://openvswitch.org

Q & A