Fulvio Risso, Matteo Bertrone, Mauricio Vasquez Bernal

Politecnico di Torino, Italy

COUPLING THE FLEXIBILITY OF OVN WITH THE EFFICIENCY OF IOVISOR: ARCHITECTURE AND DEMO
Datacenter networking nowadays requires a mix of different technologies
  – Linuxbridge, OvS, OpenFlow, Linux containers/Docker with native services (e.g., DHCP), …

Difficult to write and deploy new functions
The idea

• OVN
  – Addresses the problem of datacenter-wise orchestration
  – Runs with different cloud management systems (OpenStack, Mesos, …)
  – Bottom-layer technologies are the usual suspects

• IOVisor
  – Enables the creation of powerful network functions
    • Fast: running in kernel
    • Dynamically injectable at run-time
    • More flexible than OpenFlow actions
• Community-driven open source project
• Provides development tools that enable the creation of modules (IOModules), which can be dynamically injected in the kernel at run-time
• IOModules can be used to build networking (e.g., network functions), security, and tracing applications
**eBPF** is a virtual machine that extends the classical BPF instruction set architecture
  - Includes just-in-time (JIT) compiler and a powerful verifier that avoids inconsistencies, safety issues and hazards
  - Now part of the Linux kernel

**An IOModule** is an *eBPF* program that performs a specific task
  - *Bridge, Router, NAT, etc.*

**IOModules** can be combined to create complex services (*service chain*)
Hover

- Userspace daemon that interacts with the Linux kernel and handles the lifecycle of IOModules
- It exposes a **REST API** front-end for dynamically loading, configuring, linking different IOModules to create a *service chain*
OpenStack - OVN architecture

- **Northbound DB**: high-level description of network services (logical switches, logical routers, etc.)
- **northd**: converts high-level descriptions from northDB to flow-like descriptions in southDB
- **Southbound DB**: logical flows and bindings tables, split per each compute node
- **ovn-controller**: executed on each hypervisor
  - Pushes the flows in OvS
  - Propagates any “physical” layer event (e.g., port down) to the upper layers
- **Open vSwitch**: in charge of the dataplane
- **Bidirectional** communication across the entire stack
IOVisor-OVN: architecture

- Starting point: keep OVN control plane, remove OvS dataplane
- IOVisor-OVN
  - **Reads** info from existing databases (NorthDB, SouthDB, ovs-localDB(s))
  - **Maps** changes (e.g., service requests) into IOModules
  - **Exploits** Hover to inject, configure and bind IOModules to network interfaces
IOVisor-OVN: internals

- **OVN Monitor**: in charge of the synchronization with OVN databases

- **Main Logic**: processing logic that reacts to DB changes and/or notification coming from hypervisors

- **Hover Ctrl**: talks with Hover daemon running on each hypervisor
OVN modifications

• Very limited
  – OVN-controller does not have to propagate commands to OvS (e.g., flow rules)
  – OVN does not have to start the OvS dataplane

• Some information are ignored
  – E.g., flow rules (that are no longer used to create bridged networks)
• Proof of concept supports **OpenStack** creating a **L2 network** on a **single compute node** through the ML2 **OVN mechanism driver**
  – Reacts to the following changes in the OVN databases: a switch, a port or a port security rule is added, modified or removed
  – When appropriate, it injects and configures the L2Switch IOModule through Hover (e.g., attaching the VM vNIC to the IOModule)
  – Handles the dynamic mapping of logical names (e.g., “port12” coming from Neutron) with actual names (e.g., “tap34” coming from the hypervisor)
IOVisor-OVN: current status (2)

- Created a minimal IOModules repository
  - L2 Switch with optional port security based on MAC and IP address
- CLI for debugging
  - Status of each IOModule on the hypervisor
  - Mapping of OVN info to IOModules
IOVisor-OVN: next steps

- Extend IOModules repository with more network functions
  - Router
  - NAT
- Extend the architecture to handle multiple hypervisors
- Investigate possible optimization strategies when distributed network functions are needed
How to try OVN-IOVisor

• Full stack deployable through DevStack
  – OpenStack, OVN*, Hover, IOVisor-OVN and all the associated dependencies
    • OVN*: vanilla OVN, with only OvS control plane (no data plane)
  – Single compute node

• More info:
  – https://github.com/netgroup-polito/iovisor-ovn/
Demo Overview

- Single node DevStack setup, with IOVisor-OVN as network backend
- Logical steps:
  - Create an L2 Network
  - Instantiate two VMs connected to that network
  - VM1 and VM2 can ping each other
  - Now, change (manually) the IP address on VM2
    - Security rule no longer appropriate
    - Ping does no longer work
Our generous sponsors

PLUMgrid

Facebook

eit Digital
Links

• IOVisor Project
  – [https://www.iovisor.org/](https://www.iovisor.org/)

• IOVisor-OVN

• Customized versions of ovs and networking-ovn
  – [https://github.com/netgroup-polito/ovs](https://github.com/netgroup-polito/ovs)
  – [https://github.com/netgroup-polito/networking-ovn](https://github.com/netgroup-polito/networking-ovn)