Traffic shaping with OVS and SDN

Ramiro Voicu
Caltech

LHCOPN/LHCONE, Berkeley, June 2015
SDN Controllers

• “Standard” SDN controller architecture
  – NB: RESTful/JSON, NETCONF, proprietary, etc
  – SB: OF, SNMP, NETCONF,
Design considerations for SDN control plane

• Even at the site/cluster level the SDN control plane *should* support fault-tolerance and resilience
  – e.g. in case one or multiple controller instances fail, the entire *local* control plane must continue to operate

• Each NE must connect to at least two controllers (redundancy)

• Scalability
SDN Controllers

• **NOX:**
  - First SDN controller
  - C++
  - New protocols development
  - Open source by Nicira in 2008

• **POX**
  - Python-based version of NOX

• **Ryu**
  - Python
  - Integration with OpenStack
  - Clustering: No SPOF using Zookeeper
ONOS: Open Network Operating System

- “A SDN network operating system for Service Providers and mission critical networks”
- Distributed core designed for High-Availability and performance (RAFT consensus algorithm)
- Developed in Java - a set of OSGi modules deployed in an OSGi container (Karaf) – similar to ODL
- Application Intents to NB which can be compiled on the fly in (Flow) Rules for SB
  - Intent: Request a service from the network without knowing how the service will be performed
ONOS: Open Network Operating System
OpenDaylight “Helium”
OpenDaylight “Helium”

- Project under Linux Foundation umbrella
- Well-established and very active community (very likely the biggest)
- Developed in Java - a set of OSGi modules deployed in an OSGi container (Karaf) – similar to ONOS
- Backed up by leading industry partners
- Distributed clustering based on Raft consensus protocol
- OpenStack integration
- “Helium” is the 2nd release of ODL and supports, apart from SDN, NV (Network Virtualization) and NFV (Network Function Virtualization)
“Open vSwitch is a production quality, multilayer virtual switch”

- OpenFlow protocol support (1.3)
- Kernel and user space forwarding engines
- Runs on all major Linux distributions used in HEP[*]
- NIC bonding
- Fine grained QoS support
  - Ingress qdisc, HFSC, HTB
- Used in a variety of hardware platforms (e.g. Pica8) and software appliances like Mininet
- Interoperates with OpenStack

- OVN (Open Virtual Network): Virtualized network “implemented on top of a tunnel-based (VXLAN, NVGRE, IPsec., etc) overlay network”

[*] For SL/CentOS/RH 6.x some patches had to be applied. Custom RPMs
OVS Open vSwitch

Performance tests

- Compared the performance of hardware versus the OVS in two cases:
  - Bridged network (the physical interface becomes a port in the OVS switch)
  - Dynamic bandwidth adjustment
Baseline performance tests (10Ge)

- Two SanyBridge machines
- 10Ge Mellanox cards ("back-to-back")
- Stock SLC 6.x kernel
- Connected via Z9K
Baseline performance – hardware throughput (single stream)

- FDT nettest (memory to memory)

- 1 TCP Stream
- 10Gbps
Baseline performance (single stream)

- FDT nettest (memory to memory)
- Line rate 10Gbps

CPU utilization receiver

95% idle

CPU utilization sender

95% idle
Baseline performance (multiple stream)

- FDT nettest (memory to memory)
- CPU Usage:
  - Receiver: ~95% Idle
  - Sender: ~92% Idle (64 streams), 90% Idle (128 streams)
- Similar results with 8, 16, 32, 64 and 128 TCP streams

**CPU utilization receiver**

95% idle

**CPU utilization sender**

93% idle
Performance tests – OVS Setup

- Same hardware
- OVS 2.3.1 on stock SLC(6) kernel
- Same eth interfaces added as OVS interfaces
  - `ovs-vsctl add port br0 eth5`
OVS performance

- FDT nettest (memory to memory)
- Line rate 10Gbps
- Slightly decreased performance for receiver

CPU utilization receiver: 95% idle
CPU utilization sender: 90% idle

128 TCP Streams
Conclusions: Baseline performance tests

- TCP tests with a range of multiple TCP streams varying from 1 up to 128
- Line rate in all the tests: 10Gbps
- CPU usage between 5% and 10% (normal scenario with <64 streams would be 95% CPU idle)
- OVS 2.3.1 with stock SL/CentOS/RH 6.x kernel
- OVS bridged interface achieved the same performance as the hardware (10Gbps)
- No CPU overhead for OVS in this scenario
OVS Dynamic bandwidth adjustment
Ingress rate-limit

- OVS ingress rate-limit
- Adjust per interface
  - Based on Linux kernel "ingress qdisc"
OVS Dynamic bandwidth adjustment
Ingress rate-limit

No policy
10Gbps

10Mbps
100Mbps
2.5Gbps
5Gbps
7.5Gbps
9Gbps
10Gbps policy
OVS Dynamic bandwidth adjustment
Ingress rate-limit – Receiver CPU

Almost the same CPU Usage as without ingress policy in place
OVS Dynamic bandwidth adjustment
Ingress rate-limit – Sender CPU

Almost the same CPU Usage as without ingress policy in place
OVS Dynamic bandwidth adjustment

- OVS egress rate-limit
- Based on Linux kernel:
  - HTB (Hierarchical Token Bucket)
  - HFSC (Hierarchical Fair-Service Curve)
OVS Dynamic bandwidth adjustment egress rate-limit

10Gbps NO policy

9Gbps policy

7Gbps

11Gbps policy

500Mbps

5Gbps

10Gbps policy

1Gbps

7.5Gbps

500Mbps

2.5Gbps

5Gbps
OVS Dynamic bandwidth adjustment egress rate-limit

Almost the same CPU Usage as without egress policy in place
OVS Dynamic bandwidth adjustment egress rate-limit - Sender CPU

Almost the same CPU Usage as without egress policy in place
Conclusions:  
OVS Dynamic bandwidth adjustment  

- Smooth egress traffic shaping up to 10Gbps, and up to 7Gbps for ingress  
- Over long RTT the ingress traffic shaping may not perform well (needs more testing), especially above 7Gbps  
- The CPU overhead is negligible when enforcing QoS  
- More testing is needed:  
  - Longer RTTs  
  - 40Ge? (are there any storage nodes with 40Ge yet)  
  - Multiple QoS queues for egress  
  - reliability over longer intervals
All NEs controlled by a redundant SDN control plane

The entire SDN resides within the network layer
OVS daemon runs on the end-host

May evolve in a site with SDN-controlled NEs

OVS is “ready” to be deployed today without any impact on the current operations
Possible OVS benefits

- The controller gets the “handle” all the way to the end-host
- Traffic shaping (egress) of outgoing flows may help performance in cases where upstream switch has smaller buffers
- A SDN controller may enforce QoS in non-OpenFlow clusters
- OVS 2.3.1 with *stock* SL/CentOS/RH 6.x kernel
- OVS bridged interface achieved the same performance as the hardware (10Gbps)
- No CPU overhead for OVS in this scenario