Agenda

• What is networking-vpp?
• Design principles
• Overall architecture
• Current feature set
• FDS/networking-vpp and OPNFV
• Thank You - OPNFV, Functest and Fuel (and Apex)
• Roadmap for 20.05
• Questions
What is networking-vpp

• FD.io / VPP is a **fast software dataplane** that can be used to speed up communications for any kind of VM or VNF.

• VPP can speed-up both East-West and North-South communications

• Networking-vpp is a project aiming at providing a **simple, robust, production grade** integration of VPP in OpenStack using ml2 interface

• Goal is to make VPP a **first class citizen component in OpenStack** for NFV and Cloud applications
VPP L2 Switching

Single core

- base (508 MACs) Throughput (Mpps): 20.52
- 10K MACs: 15.12
- 100K MACs: 12.34
- 1M MACs: 11.42

- Throughput (Mpps)

Intel CascadeLake@2.3GHz (Xeon 6252N), HyperThread on, TurboBoost off
VPP L3 Routing

Single Core

- base (508 FIBs): Throughput (Mpps) = 19.32
- 20K FIBs: Throughput (Mpps) = 18.43
- 200K FIBs: Throughput (Mpps) = 18.54
- 2M FIBs: Throughput (Mpps) = 17.11

Intel CascadeLake@2.3GHz (Xeon 6252N), HyperThread on, TurboBoost off
VPP vhostuser (Phy-Virtual-Phy aka PVP)

Throughput (Mpps)

- `eth-l2xc`: 4,163
- `eth-l2bd`: 3,167
- `dot1q-l2bd`: 2,552

Intel **CascadeLake@2.3GHz** (Xeon 6252N), HyperThread on, TurboBoost off
Networking-vpp: Design Principles

- Main design goals are: *simplicity, robustness, scalability*

- Efficient management communications
  - All communication is asynchronous
  - All communication is REST based

- Robustness
  - Built for failure – if a cloud runs long enough, *everything* will happen eventually
  - All modules are unit and system tested

- Code is small and easy to understand (no spaghetti/legacy code)
Networking-vpp, what is your problem?

• You have a controller and you tell it to do something
• It talks to a device to get the job done
  • Did the message even get sent, or did the controller crash first? Does the controller believe it sent the message when it restarts?
  • Did the message get sent and the controller crash before it got a reply?
  • Did it get a reply but crash before it processed it?
  • If the message was sent and you got a reply, did the device get programmed?
  • If the message was sent and you didn’t get a reply, did the device get programmed?
Networking-vpp, what is your problem?

• If you give a device a list of jobs to do, it’s really hard to make sure it gets them all and acts on them.
• If you give a device a description of the state you want it to get to, the task can be made much easier.
Networking-vpp: overall architecture

- Compute Node
  - VM
  - VPP
  - VPP Agent
  - vhostuser
  - dpdk

- Neutron Server
  - ML2 VPP Mechanism Driver
  - journaling
  - MariaDB

- etcd

- VLAN / Flat Network
Networking-vpp: current feature set

- Network types
  - VLAN: supported since version 16.09
  - VXLAN-GPE: supported since version 17.04

- Port types
  - VM connectivity done using fast `vhostuser` interfaces
  - TAP interfaces for services such as DHCP
  - GSO support for increased endpoint performance

- Security
  - Security-groups based on VPP stateful ACLs
  - Port Security can be disabled for true fastpath
  - Role Based Access Control and secure TLS connections for etcd

- Layer 3 Networking
  - North-South Floating IP, SNAT
  - East-West Internal Gateway

- Robustness
  - If Neutron commits to it, it will happen
  - Component state resync in case of failure: recovers from restart of Neutron, the agent and VPP
  - LACP bonding for uplinks, ECMP for L3

- TaaS
  - Supported since version 18.10
  - ERSPAN support since version 19.08.1

- Python3
  - Support since version 19.01
  - Python3 only since version 20.01

- API versioning
  - Supported from version 20.01 onwards
  - Check against installed API signature at agent startup
  - Only whitelisted APIs allowed during runtime
FDS/networking-vpp and OPNFV

• Networking-vpp is part of OPNFV/FastDataStacks - https://wiki.opnfv.org/display/fds

• Networking-vpp is included in os-nosdn-fdio-[noha|ha] scenario

• Initial engagement (2016):
  • Work to integrate with Apex installer started from Colorado 1.0
  • Was in Colorado 3.0 that we managed to pass release criteria tests and the os-nosdn-fdio-noha scenario made it into OPNFV
OPNFV FDS/ os-nosdn-fdio-[noha|ha] overall architecture (Apex/Fuel)
Timeline

• Apex/RH
  • Colorado – 12/2016
    • passed release criteria tests
    • only non-HA scenario (os-nosdn-fdio-noha)
  • Danube – 04/2017
    • HA scenario (os-nosdn-fdio-ha)
  • Euphrates – 10/2017
    • Bugfix release
  • Fraser – 05/2018
    • L3 integration
  • Gambia – N/A -> Last available release

• Fuel
  • Hunter – 05/2019
    • L3 integration
    • non-HA scenario
  • Iruya – 01/2020
    • Python3
  • Jerma – sometimes in 2020
Thank You - OPNFV, Functest, Fuel (and Apex)

• Truly symbiotic relationship
• OPNFV gives us
  • A production like environment for testing in addition to devstack based testing
  • Access to early upstream releases / an early warning system
  • Ability to catch lots of issues during OPNFV release testing
    • Manual as well as Functest
    • Eg., Nova live migration, Trunk Port, L3, NAT and so on
• We contributed by finding & reporting issues across different projects – Apex, Fuel, Functest, snaps, domino, orchestra, cloudify_ims
  • Eg., SNAPS-185, FUNCTEST-970, APEX-468 etc
An example – Fraser release

- networking-vpp = 15 bugs found
- VPP = 4 bugs found
- Apex = 11 bugs found
- Misc = 7 bugs found

**NOTE:** Functest helped catch close to 40% of bugs in VPP/networking-vpp
Bugs found in Fraser: networking-vpp + VPP Analysis

Where were the bugs?

- core: 5
- L3/HA: 1
- L3/NAT: 3
- L3/FIP: 4
- L3: 2
- VPP: 4

How were they found?

- manual: 12
- functest: 7
Latest status – Iruya (Jan 2020)

• Installer = Fuel only
• VPP/networking-vpp version = 20.01
• OpenStack version = Stein
• Ubuntu Bionic
• Completely Python3 (Python 3.6)
Networking-vpp: Roadmap / next steps

Next version will be networking-vpp 20.05
  • https://launchpad.net/networking-vpp

• Bulk API calls
  • Speed up reconfiguration after restart
  • Support for bulk programming of ACLs using VPP async APIs

• Network APIs for new network types
  • Provide interface to easily add new overlay types

• TaaS/ERSPAN
  • Push changes upstream: ERSPAN APIs for OpenStack

• Testing, testing, testing
  • Support HA scenario for Fuel