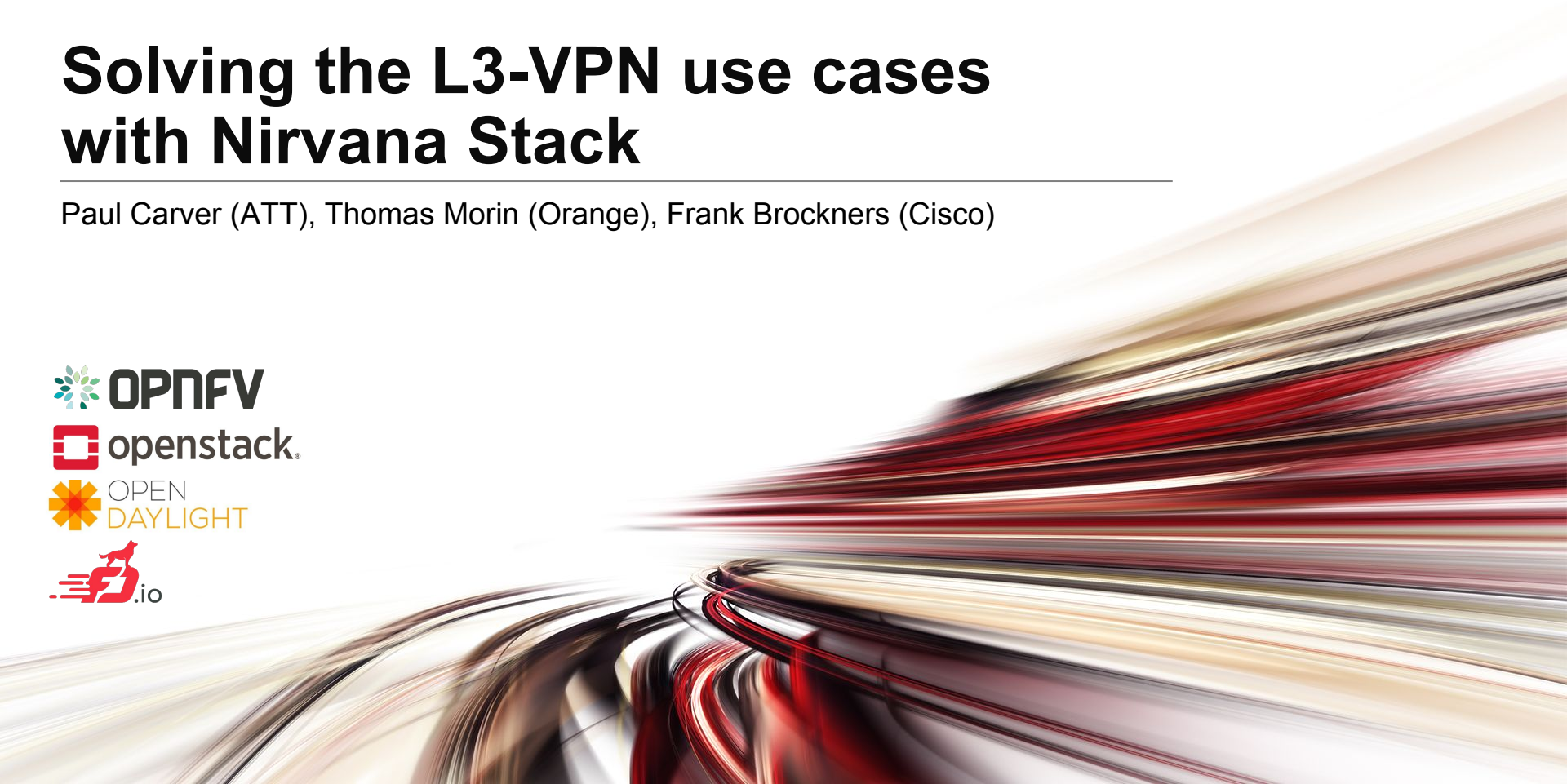
Solving the L3-VPN use cases with Nirvana Stack

Paul Carver (ATT), Thomas Morin (Orange), Frank Brockners (Cisco)



Solving the L3-VPN use cases with Nirvana Stack

- Overview of L3VPN use-cases
- Current state of components for a L3VPN solution (OpenStack Neutron, OpenDaylight and OPNFV)
- Evolving L3VPN solution stacks to include FD.io/VPP using the Nirvana Stack approach

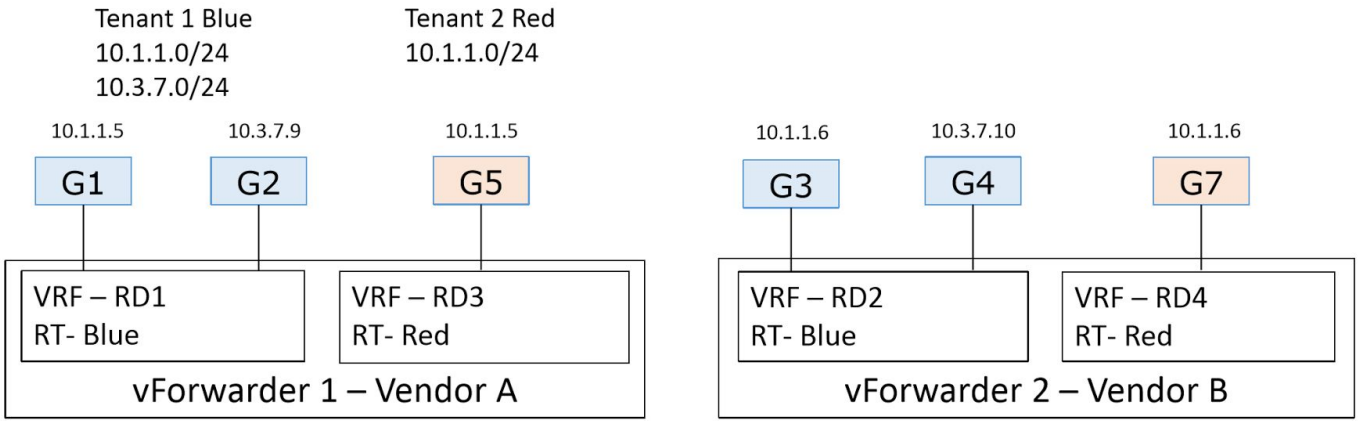
L3VPN Use-Cases

L3VPN Use-Cases

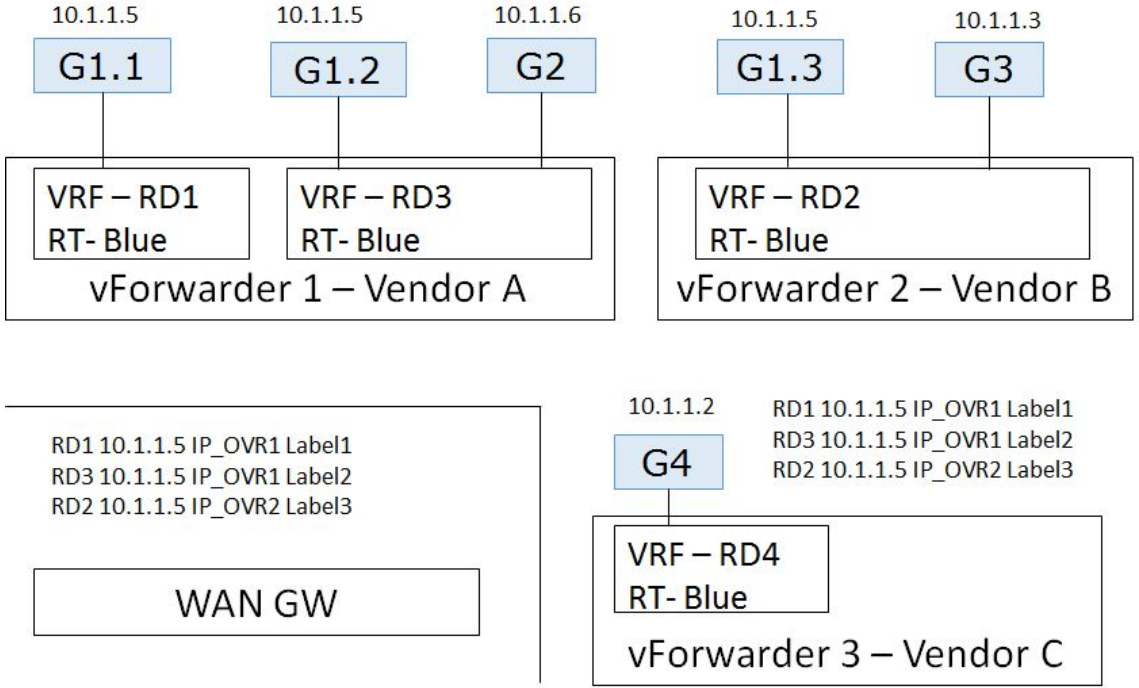
- Any-to-Any Case
- ECM Load splitting case
- Hub & Spoke case

Any-to-Any Case

- Multi-tenancy
- Segregation of Traffic
- Overlapping Address Space
- Any to any connectivity within VPN



ECMP Load Splitting Case - AnyCast



Tenant 1 Blue
10.1.1.0/24

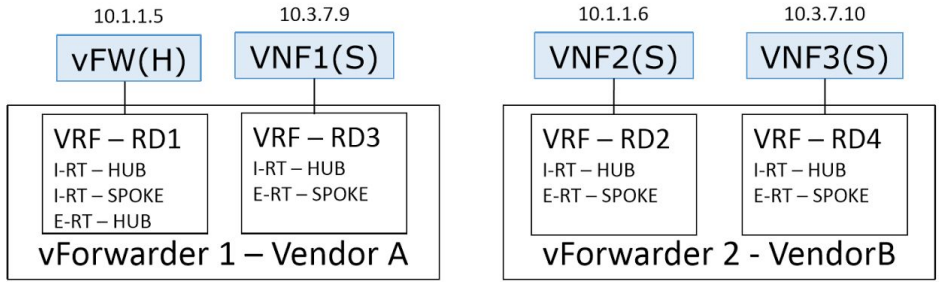
Multiple instances of a VNF are reachable through the same IP.

For example, Traffic to 10.1.1.5 can load split among 3 VNFs (G1.1, G1.2, G1.3)

Need separate RD for anycast end point to segregate traffic

Hub and Spoke Case

Tenant 1 Blue
10.1.1.0/24
10.3.7.0/24



G1 Hub VRF
RD1 10.1.1.5 IP_OVR1 Label1
RD1 0/0 IP_OVR1 Label1
Label 1 Local IF (10.1.1.5)
RD3 10.3.7.9 IP_OVR1 Label2
RD2 10.1.1.6 IP_OVR2 Label3
RD4 10.3.7.10 IP_OVR2 Label3

G2 Spoke VRF
RD1 0/0 IP_OVR1 Label1
RD3 10.3.7.9 IP_OVR1 Label2

VNFs can not communicate directly with one another. Need to go through FW.

VNF (spoke)
- exports spoke routes
- imports only hub routes.

FW (hub)
- exports hub routes
- imports and exports spoke routes

Basic building block of service chaining

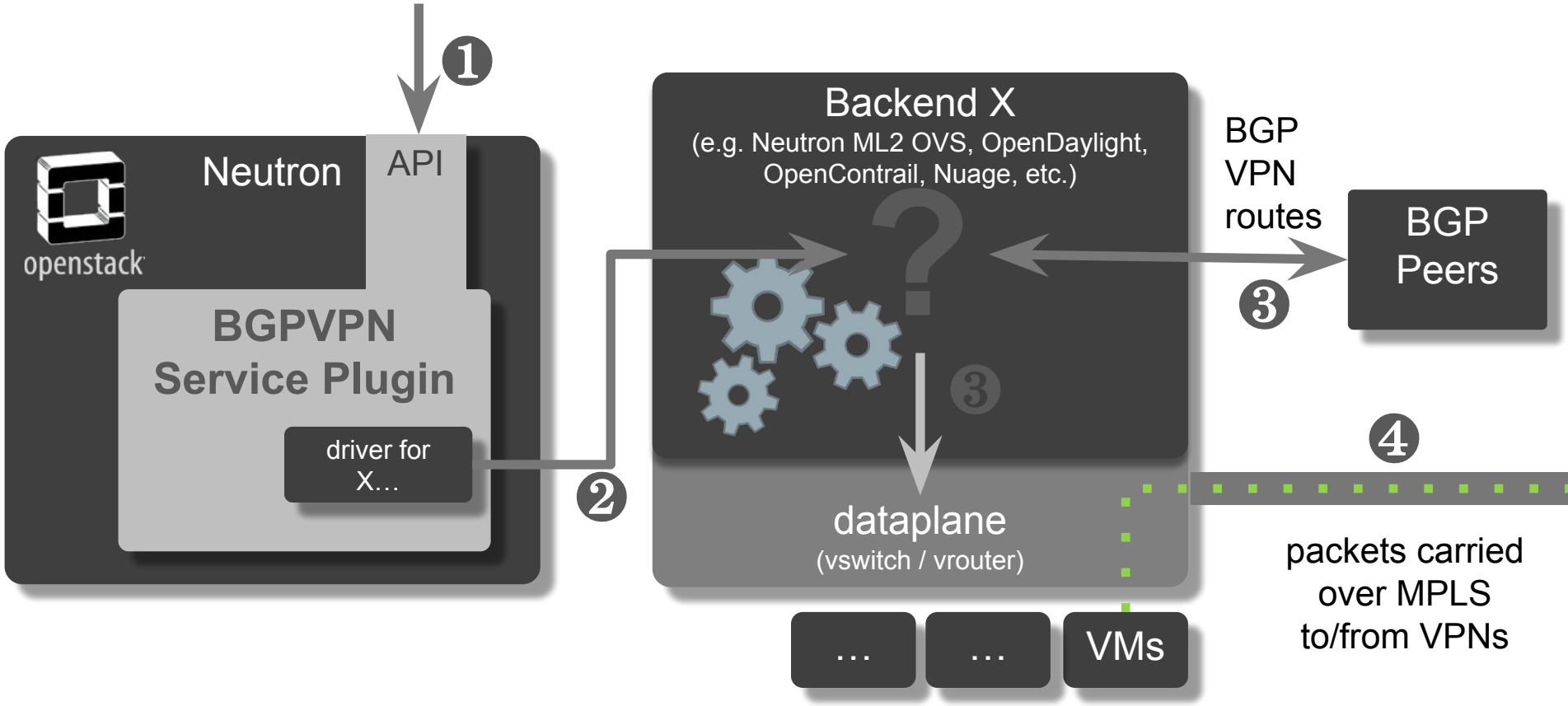
Current state of L3VPN solutions (OpenStack, OpenDaylight and OPNFV)

Emergence of a Neutron API for BGP VPN Interconnections

- Back in 2015: a variety of SDN controllers,
  each with their API to configure things related to BGP VPNs
  - OpenDaylight
  - OpenContrail
  - Nuage Networks
  - Cisco VTS
  - …
- Need for a single multi-tenant API to let tenants/orchestrator drive BGP VPN connectivity without being coupled to a specific backend

=> June 2015: Neutron BGP VPN Interconnection service
(networking-bgpvpn Neutron Stadium project)

Neutron BGP VPN Interconnection service



Status of Neutron BGP VPN Interconnection service

- Drivers for Neutron ML2 OVS and SDN controllers supporting the feature
- API to interconnect with both IP VPNs and EVPN/VXLAN
- Feature coverage addressing key aspects of the use-cases described here
  - including anycast and hub'n'spoke
  - without per Neutron port granularity, nor static routes
- Planned API evolutions on the radar:
  - Port associations
  - Static routes
  - Control of BGP local_pref and communities

Neutron BGP VPN API and the controller NBI, discussion

- “Common API + driver” model allowing us to use our stack of choice, without user/orchestrator visible changes
  - migrate toward a Nirvana stack tomorrow
  - migrate to whatever is the new Nirvana three years down the road ?

- Open question
  - How can we facilitate migrations ?
  - Allowing simultaneous use of multiple backend drivers ?

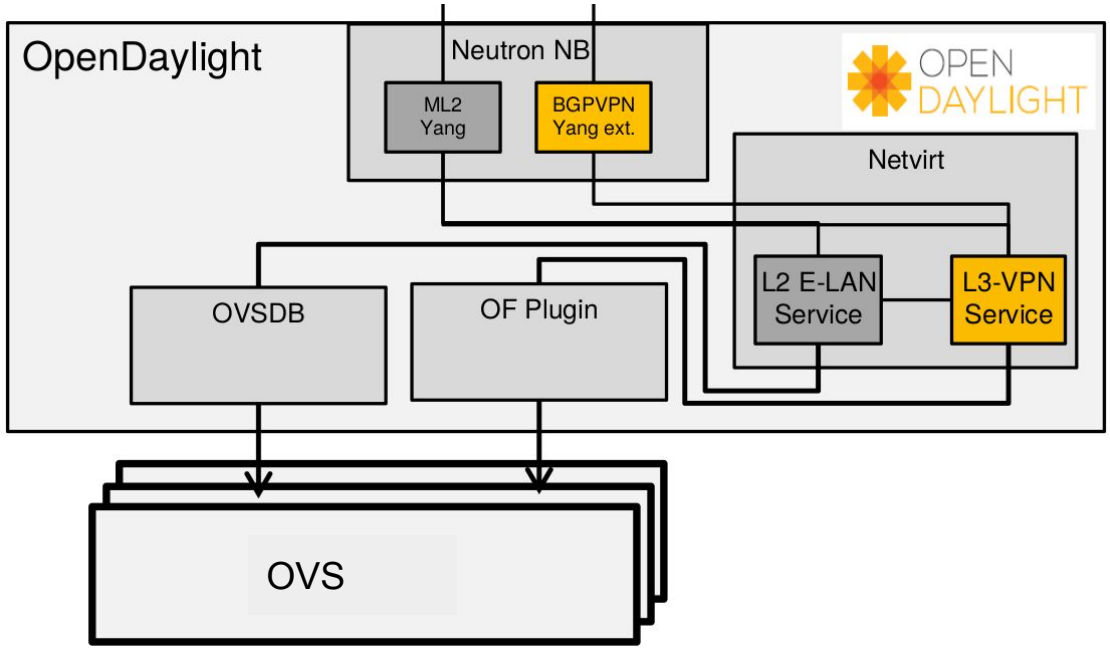
Neutron BGP VPN and OPNFV

- Neutron networking-bgpvpn project has a peer project in OPNFV: ‘SDNVPN’
- In OPNFV, the SDNVPN project aims at integrating a complete stack for BGP VPNs:
  - Currently focusing on cases where an SDN controller is used
  - Integration with installers:
    - Fuel (Mirantis) and TripleO/Apex (RedHat)
  - Provides deployment scenarios derived from odl_l3, both HA and non-HA

State of BGP VPN implementation in ODL

BGP VPN has been implemented in ODL Netvirt:

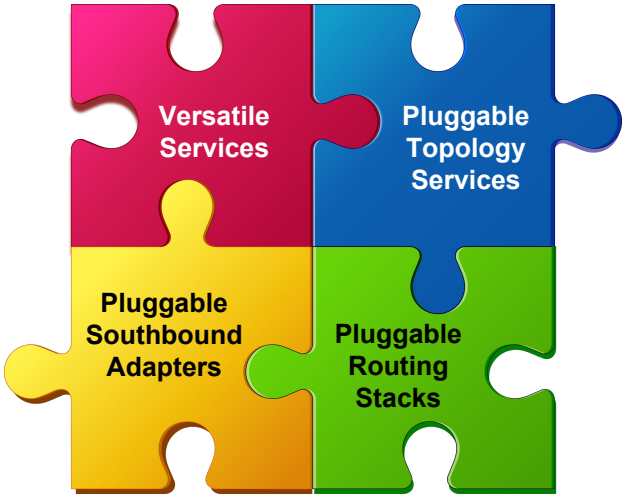
- Leverages Quagga BGP implementation for route exchange
- OVS/OVS-DPDK as the dataplane



Strawman: Evolving L3VPN solution stacks to include FD.io/VPP using the Nirvana-Stack approach

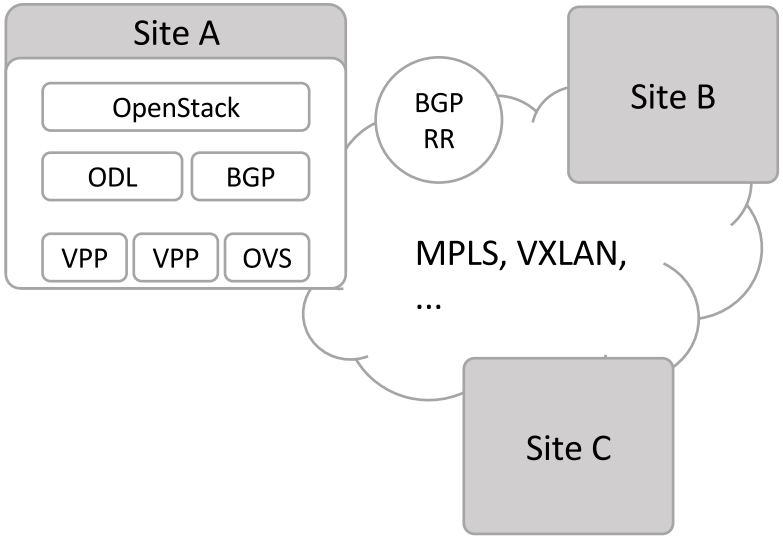
Recap from this morning: Towards the Nirvana Stack
Integrated while modular approach in OpenDaylight

- Pluggable & Modular Architecture
  - Services decoupled from forwarding technology
    - Comprehensive set of Services (leverage NetVirt)
    - Modular and pluggable southbound adapters (leverage GBP)
  - BGP routing stacks: ODL BGP, Quagga, ..
  - Topology service: LISP, VBD, ..
- Model-driven northbound API



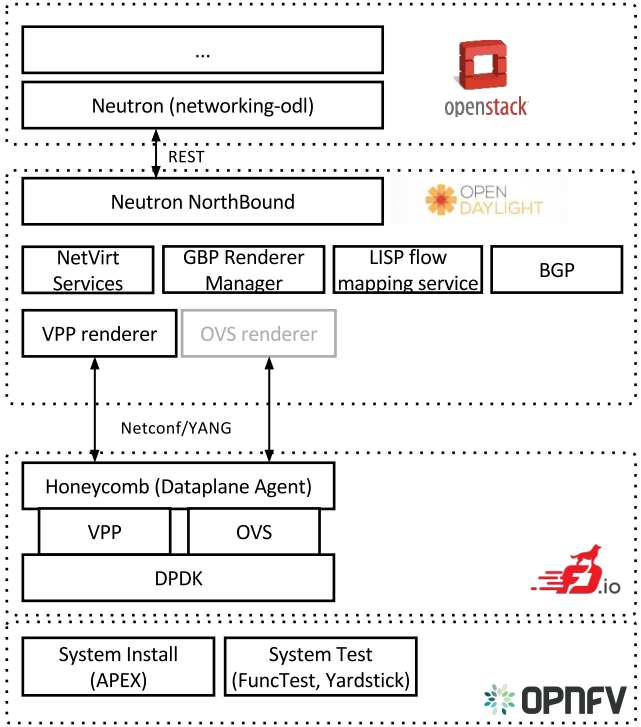
Towards L3VPN with a NirvanaStack approach: Key Design Considerations

- Sites interconnected via flexible tunnel technology (MPLS, MPLSoGRE, VXLAN,...) for “north-south” traffic
  - MP-BGP for routing per VRF
- Individual sites to implement fully distributed routing (i.e. DVR)
  - Every forwarder serves as a L3-router
- Converged Network Control solution
  - ODL GroupBasedPolicy + ODL Netvirt
  - Pluggable BGP stack (e.g. ODL BGP) – integrated or associated with Controller
  - Support for multiple forwarders (SW + HW), incl. FD.io/VPP

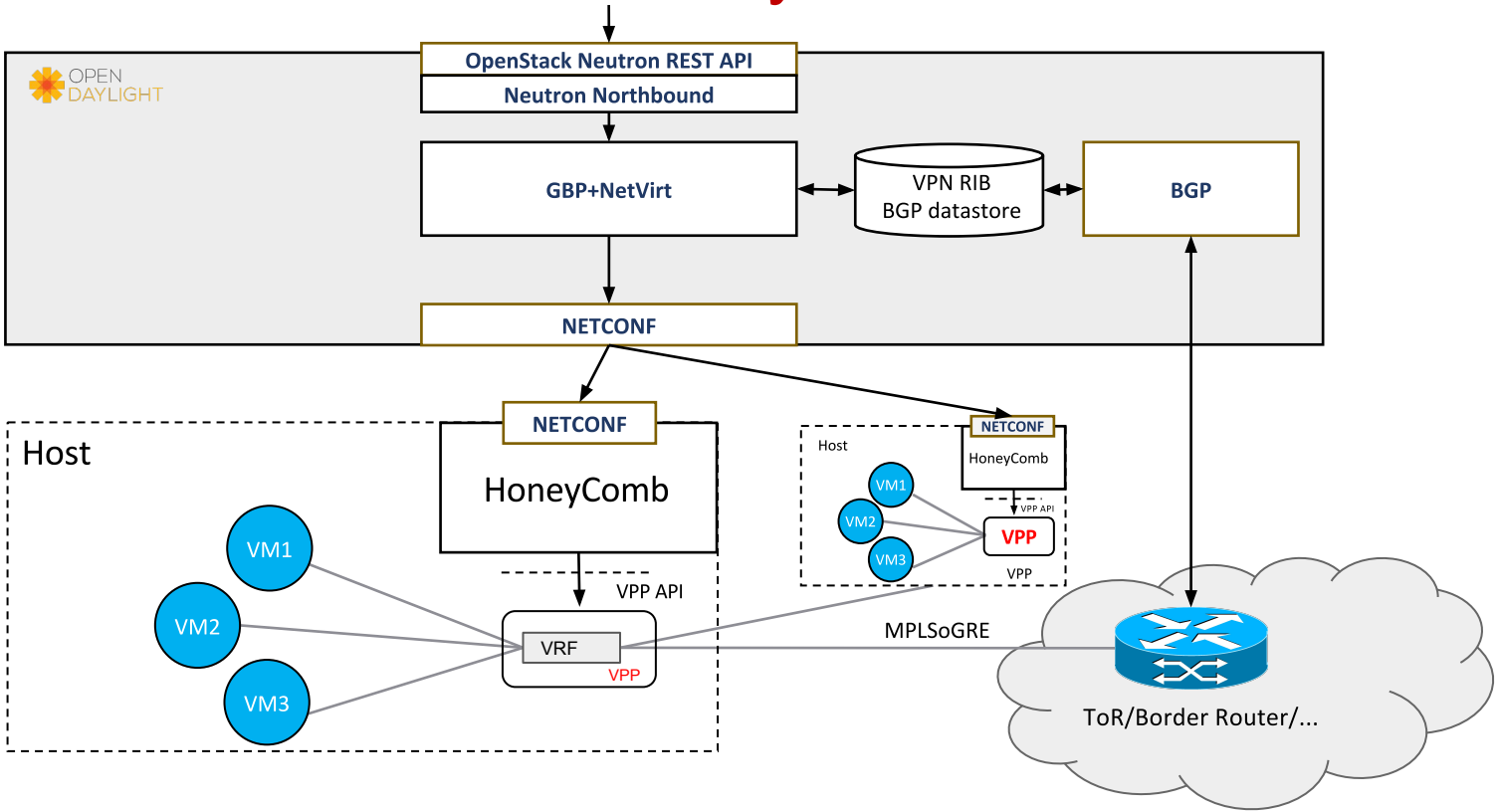


Towards L3VPN with a NirvanaStack approach: Stack Composition

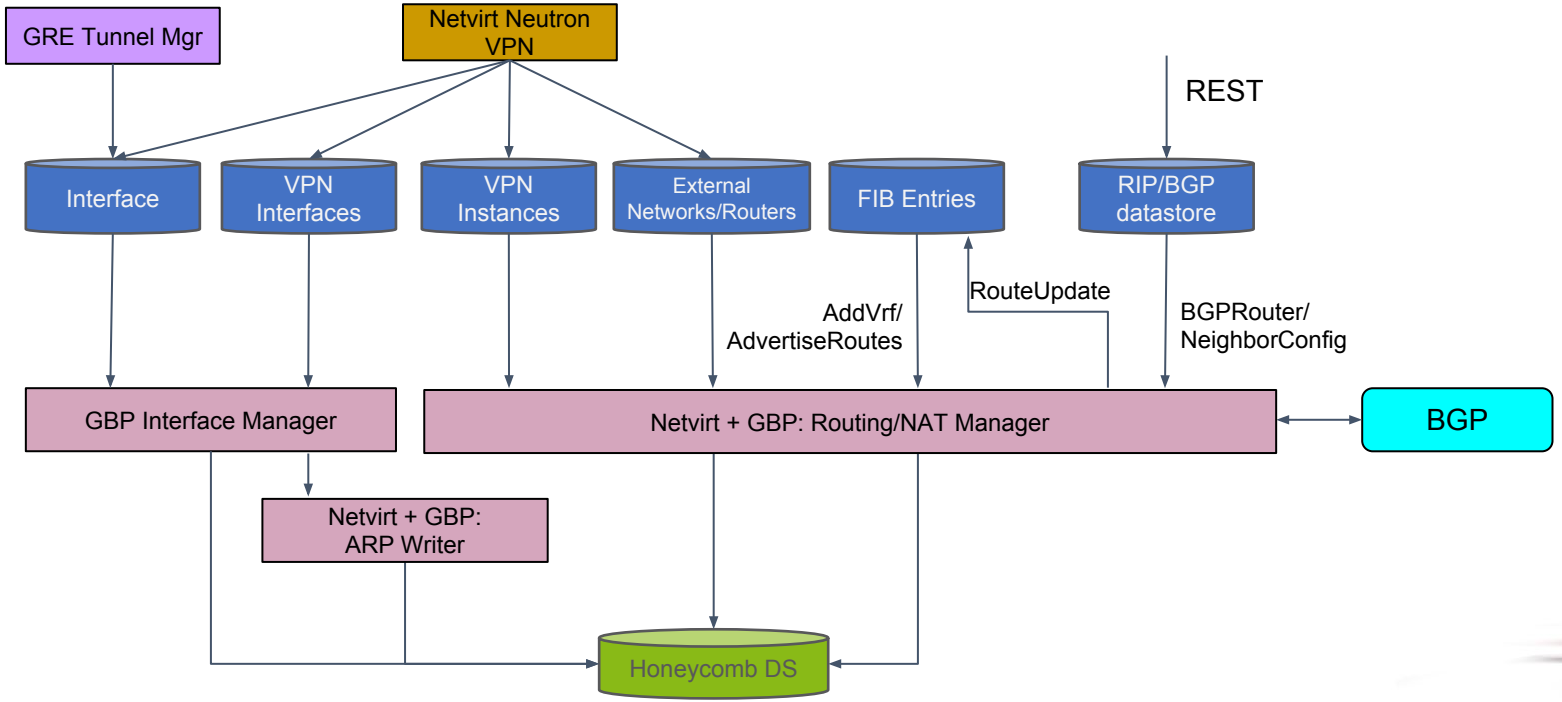
- Converged Network Control Solution
  - ODL NetVirt Service Control
  - ODL BGP stack (inter-DC traffic)
  - ODL GBP for forwarder control
  - ODL VBD/LISP flow mapping service/.. (intra-DC traffic)
- Converged Dataplane Control
  - HoneyComb for VPP, OVS control
  - Netconf/YANG as Dataplane management protocol
- OPNFV for automated system installation and testing
  - APEX (TripleO) installer, ..



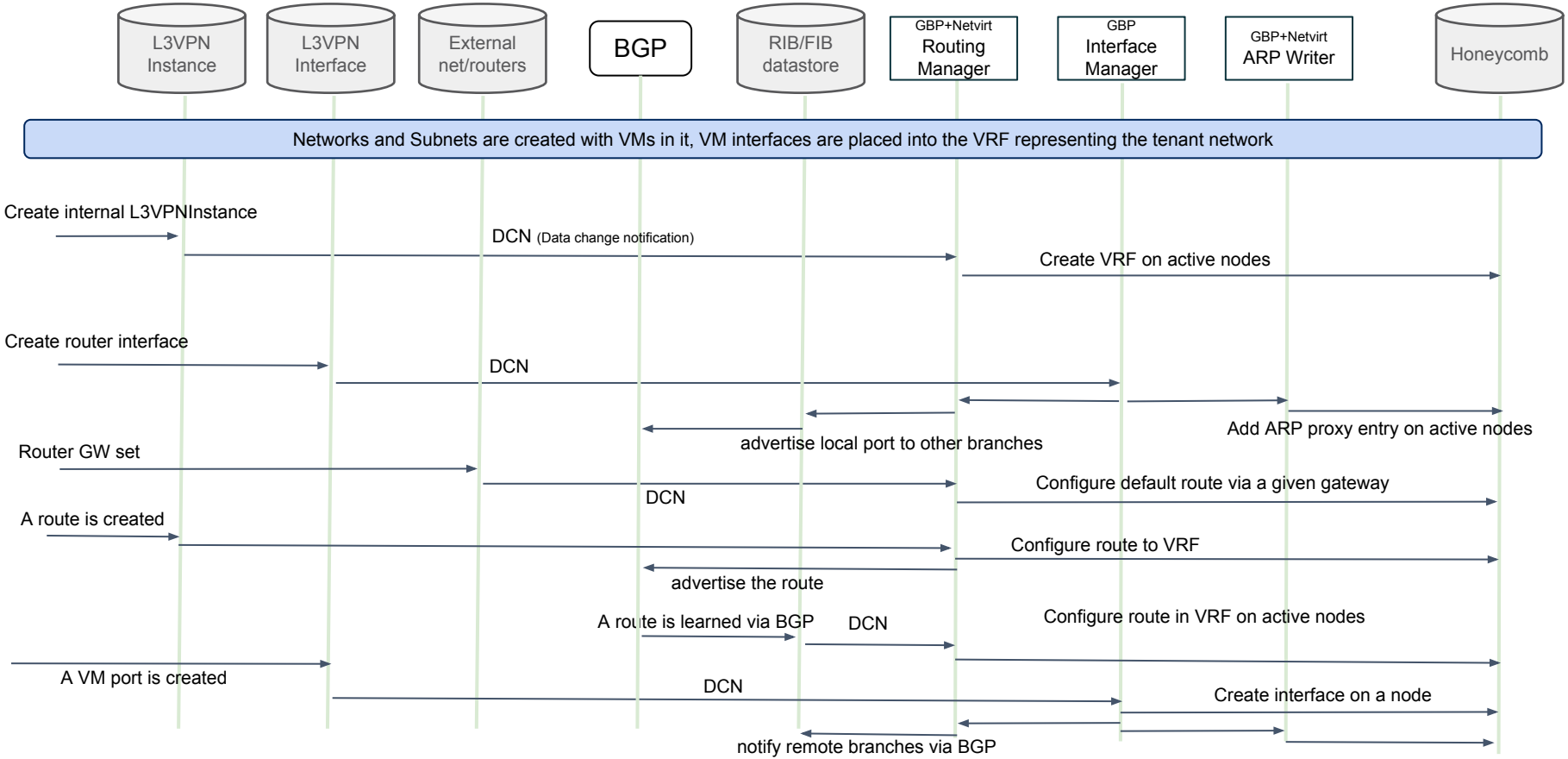
Key Design Principles:
North-South Connectivity – BGP controlled



Evolved L3VPN Service - Architecture



Evolved L3VPN Service - Call Flow



OPNFV openstack. OPEN DAYLIGHT .io

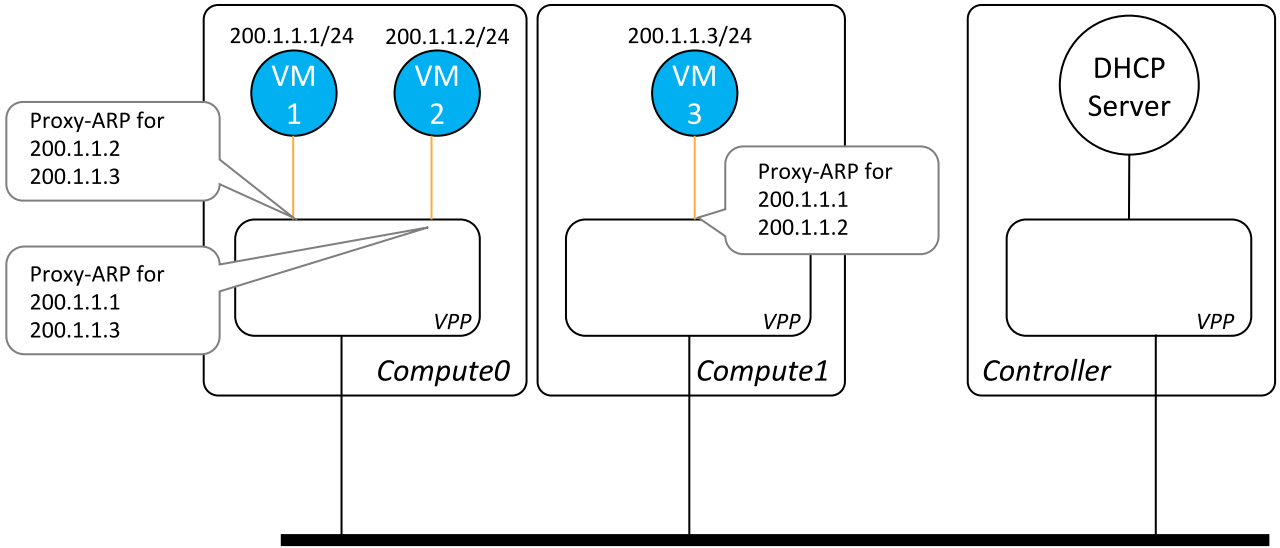
Thank you

Backup

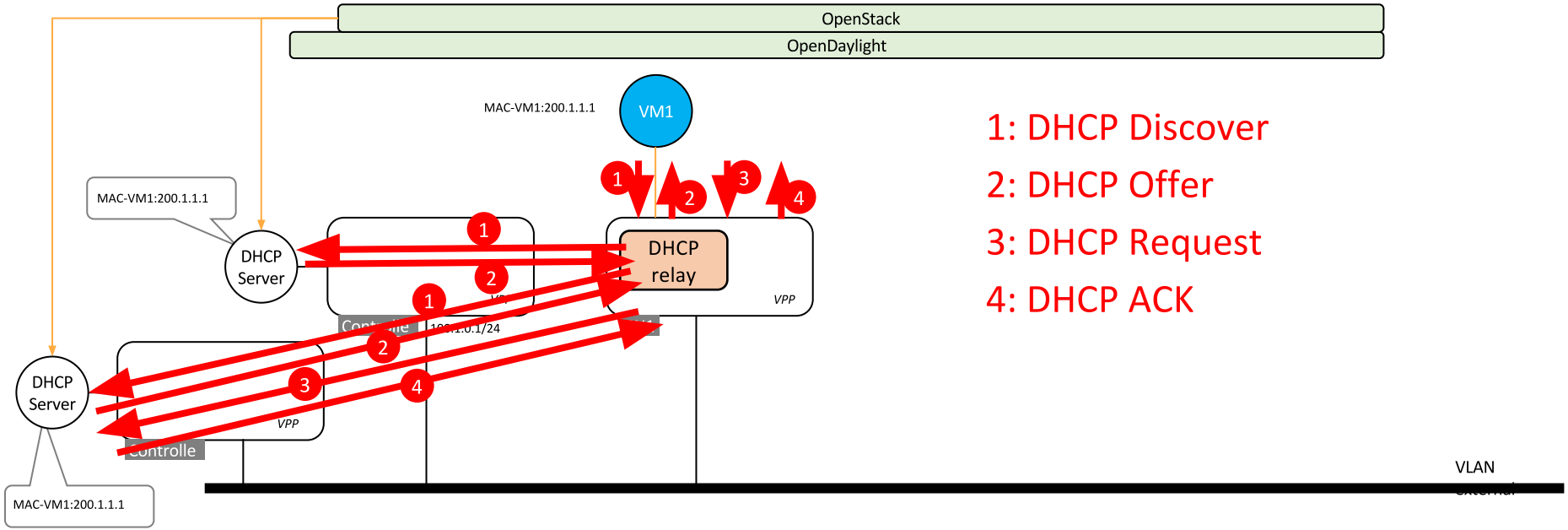
Key Design Principles:
Pure Layer 3 connectivity within a tenant; No Bridges Anywhere

Pure-Layer3

- No broadcasts
- Addresses can be aggregated
- Single end-point addressing scheme



IP Address Assignment – OpenStack HA setup

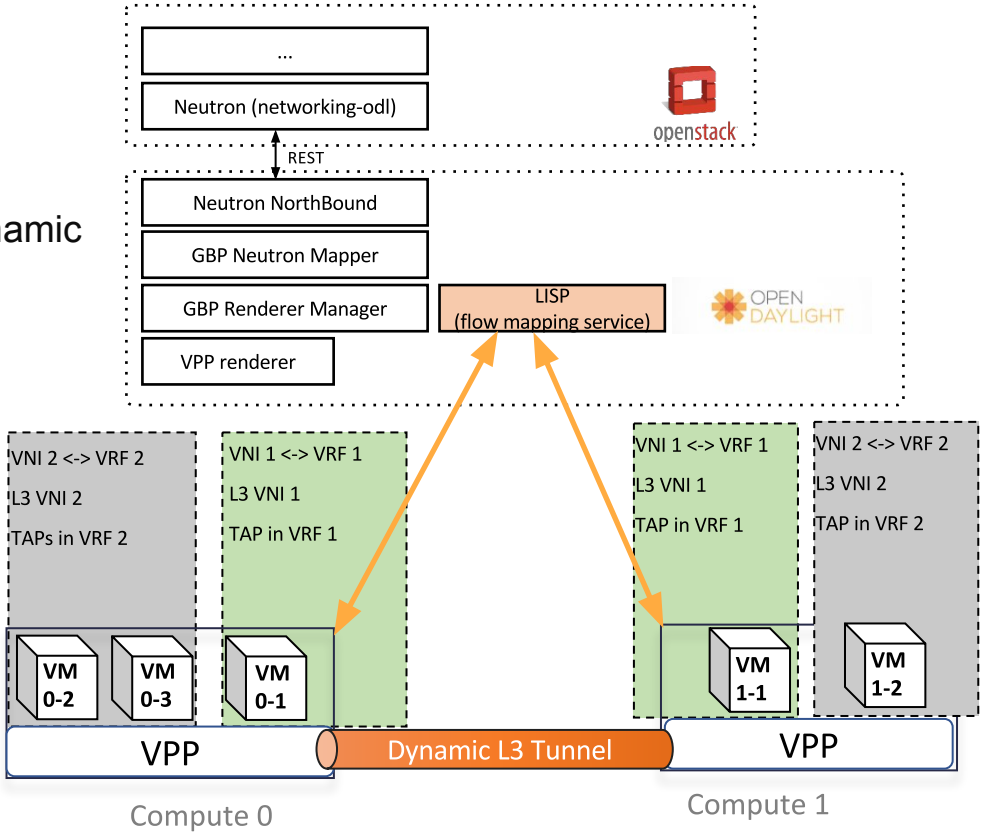


IP-MAC bindings are controlled through OpenStack, i.e. all DHCP servers are configured the same and manage the same address pool.

Key Design Principles: Static or Dynamic Tunneling for Intra-DC (east-west)

- Static tunnel config for intra-DC traffic, e.g. via VBD topology manager
- Dynamic tunnel config for intra-DC traffic: Leverage LISP Flow-Mapping Service for dynamic topology setup within a tenant network

Host	Num of VM	VM Name	VM PORT IP
Compute 0	3	VM0-1(Tenant 1)	10.11.12.2/24
		VM0-2(Tenant 2)	10.11.12.2/24
		VM0-3(Tenant 2)	10.11.12.3/24
Compute 1	2	VM1-1 (Tenant 1)	10.11.13.2/24
		VM1-2(Tenant 2)	10.11.12.4/24



LISP for Dynamic Tunneling

Host	Num of VM	VM Name	VM PORT IP
Compute 0	3	VM0-1(Tenant 1)	10.11.12.2/24
		VM0-2(Tenant 2)	10.11.12.2/24
		VM0-3(Tenant 2)	10.11.12.3/24
Compute 1	2	VM1-1 (Tenant 1)	10.11.13.2/24
		VM1-2 (Tenant 2)	10.11.12.4/24