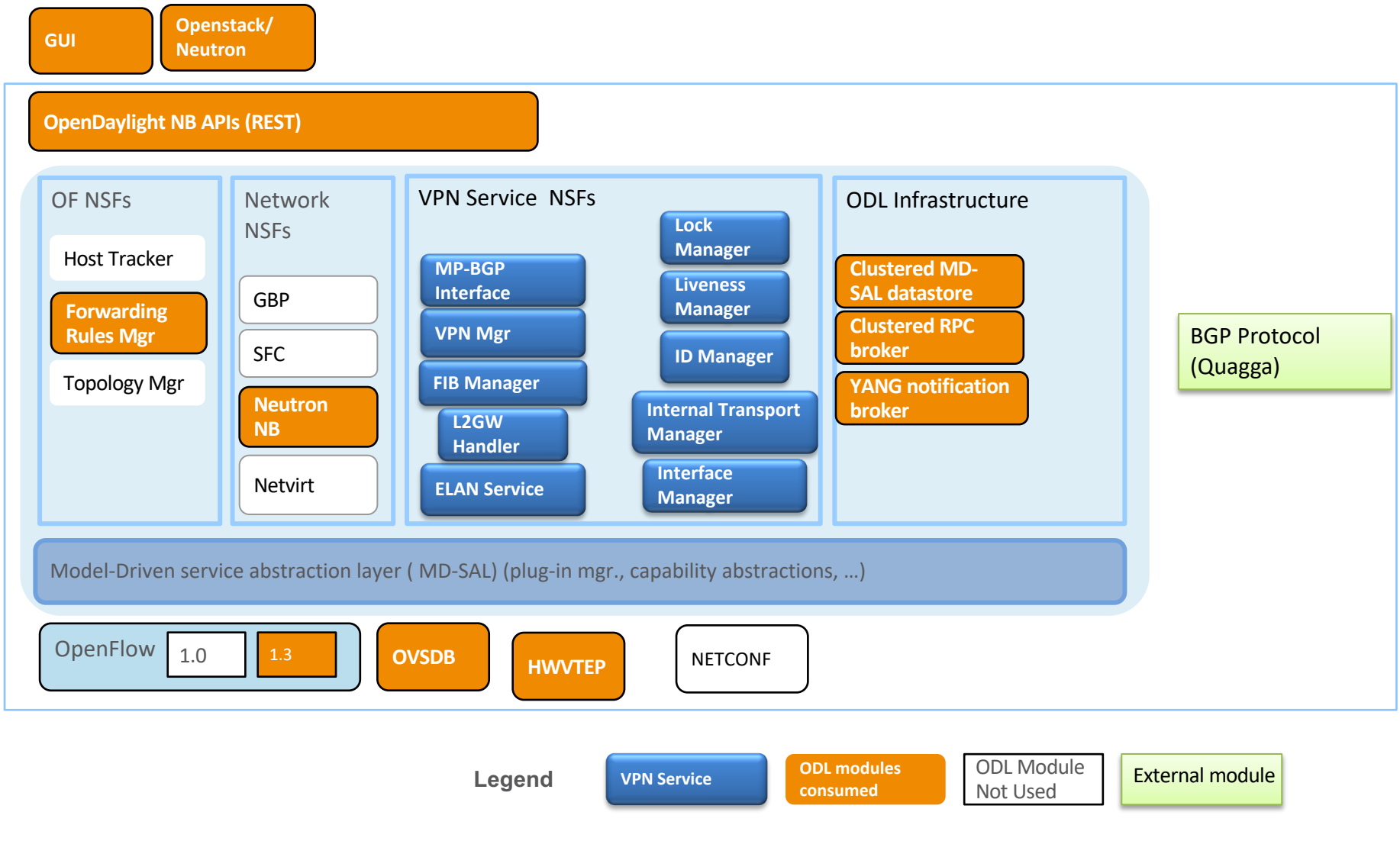
NEW NETVIRT SERVICE INTRODUCTION

Daya Kamath, Ericsson
Daya_k@yahoo.com, dayavanti.gopal.kamath@ericsson.com
Vivekanandan Narasimhan, Ericsson
n.vivekanandan@ericsson.com
Periyasamy Palanisamy
Periyasamy.palanisamy@ericsson.com

May 24 2016

VPN Service components in ODL (Be)



VPN Service-Netvirt Integration

- GOALS
  - A common service pipeline between VPN project and OVSDB - NetVirt, with shared development on a common set of services
  - Achieve maximum leverage of code, test, and developer resources
  - A modular framework of network applications and device renderers that makes addition of new functions efficient
  - Co-existence and cooperation between different services in running systems, with minimal design coordination and hard-coded dependencies

Service component comparison

Netvirt	VPN
Non MD-SAL , non-GENIUS, stateless architecture	MD-SAL and GENIUS based arch
Kernel and DPDK based OVS	Arch is not tied to DPDK or Kernel based OVS
-	GLUON compatibility through GENIUS, via SHIM layer between NN, VPN service
L2 over OVS, HWVTEP	L2 with **learning** over OVS, HWVTEP, trunk and transparent vlans
L3 over OVS	L3 over OVS with **BGP VPN support**
SFC integration	
DHCP based on Neutron agent, has metadata support	Built-in lightweight DHCP proxy for Neutron
DNAT available, SNAT WIP based on conn-track	Pure OF based DNAT, SNAT with controller assist (no ICMP support)
Distributed ARP proxy within ingress compute	Flood and punt logic for ARPs, updates VPN prefixes
10 tuple based ACL support	
LBaaS	-
IPv6 control plane (needs pipeline integration)	IPv6 (needs pipeline integration, and VPN support)
QoS (SB done, openstack integration pending)	
Conn-track based security groups	
VLAN provider type	

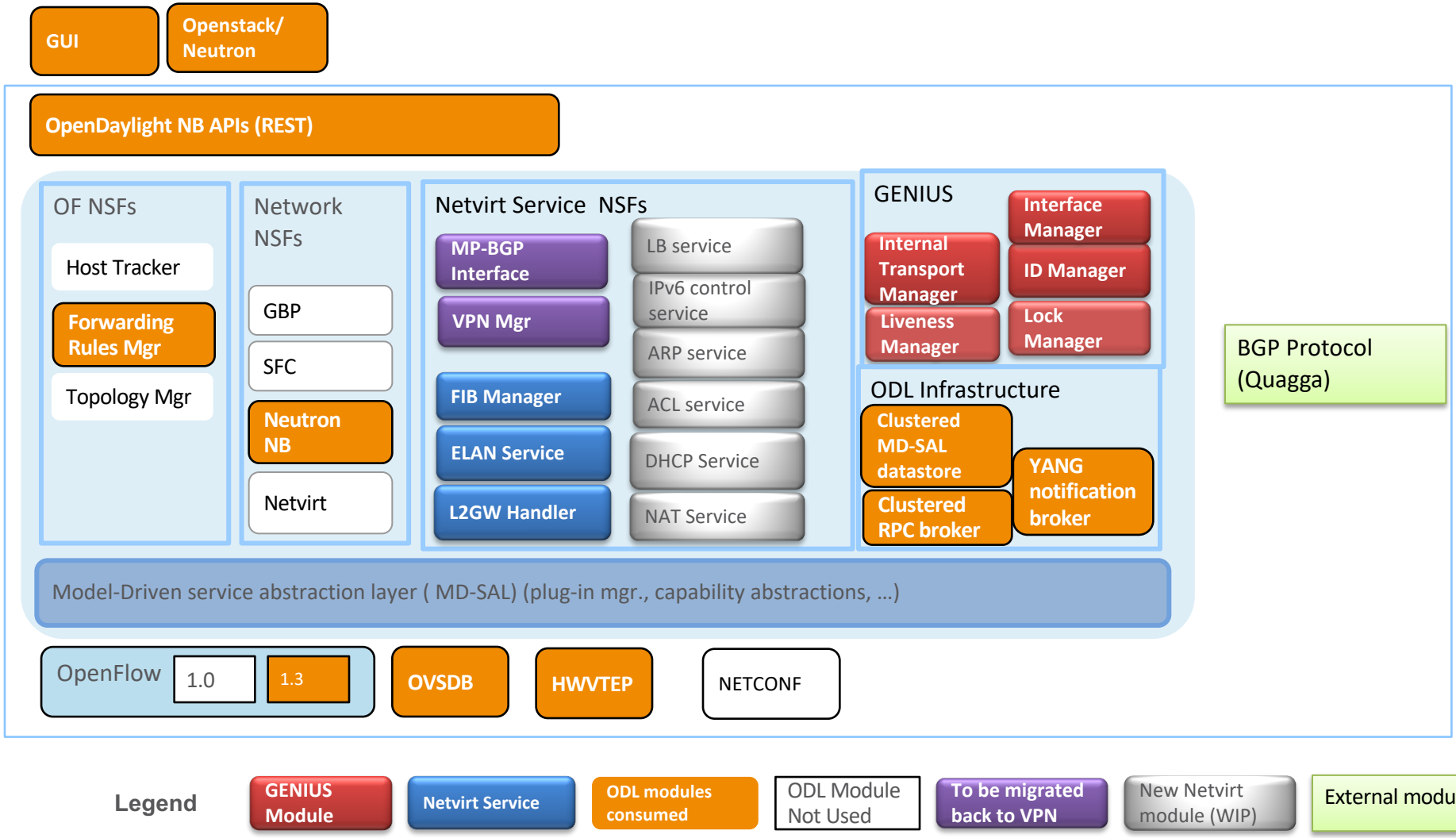
Community Direction

- Incorporate VPN service project forwarding services into Netvirt project
- Work on building feature parity between legacy Netvirt functions and VPN service functions in the new Netvirt community repository during Boron cycle
- Migrate VPN service control components back to VPN project
- Use GENIUS project framework for each sub-function to enable hybrid app co-existence

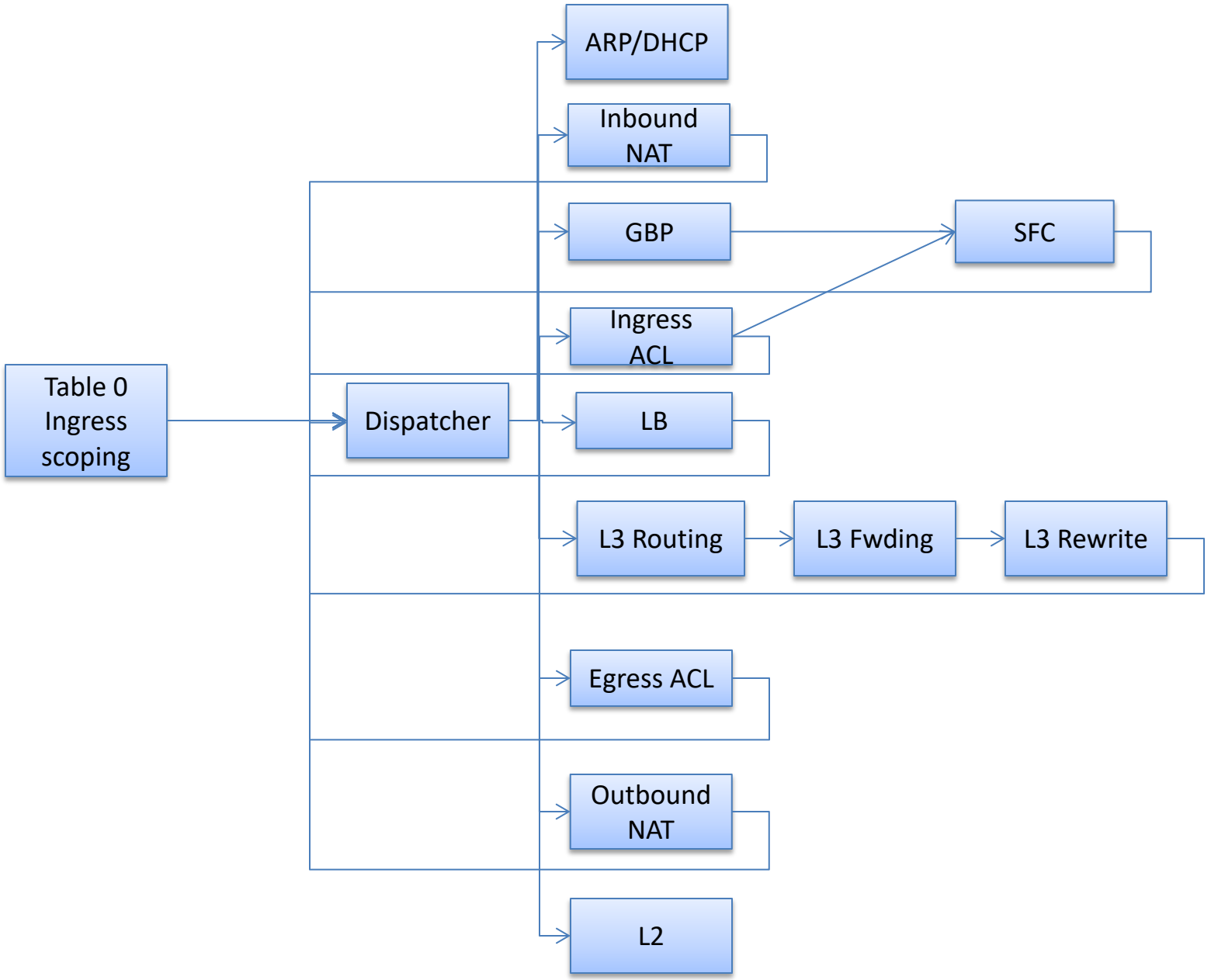
Current Status (Jun 20 2016)

- All code from VPN service migrated to Netvirt project / GENIUS project
  - Phase 1- modules migrated from VPN service to Netvirt
  - Downstream validation exercise to ensure Phase 1 sanity successful
- Netvirt services updated to use GENIUS project interfaces
  - Validation exercise to ensure Phase 2 sanity successful
- Feature parity work in progress
  - Security groups
  - DHCP service
  - ARP service
  - Deployment and testing
  - IPv6 integration into pipeline

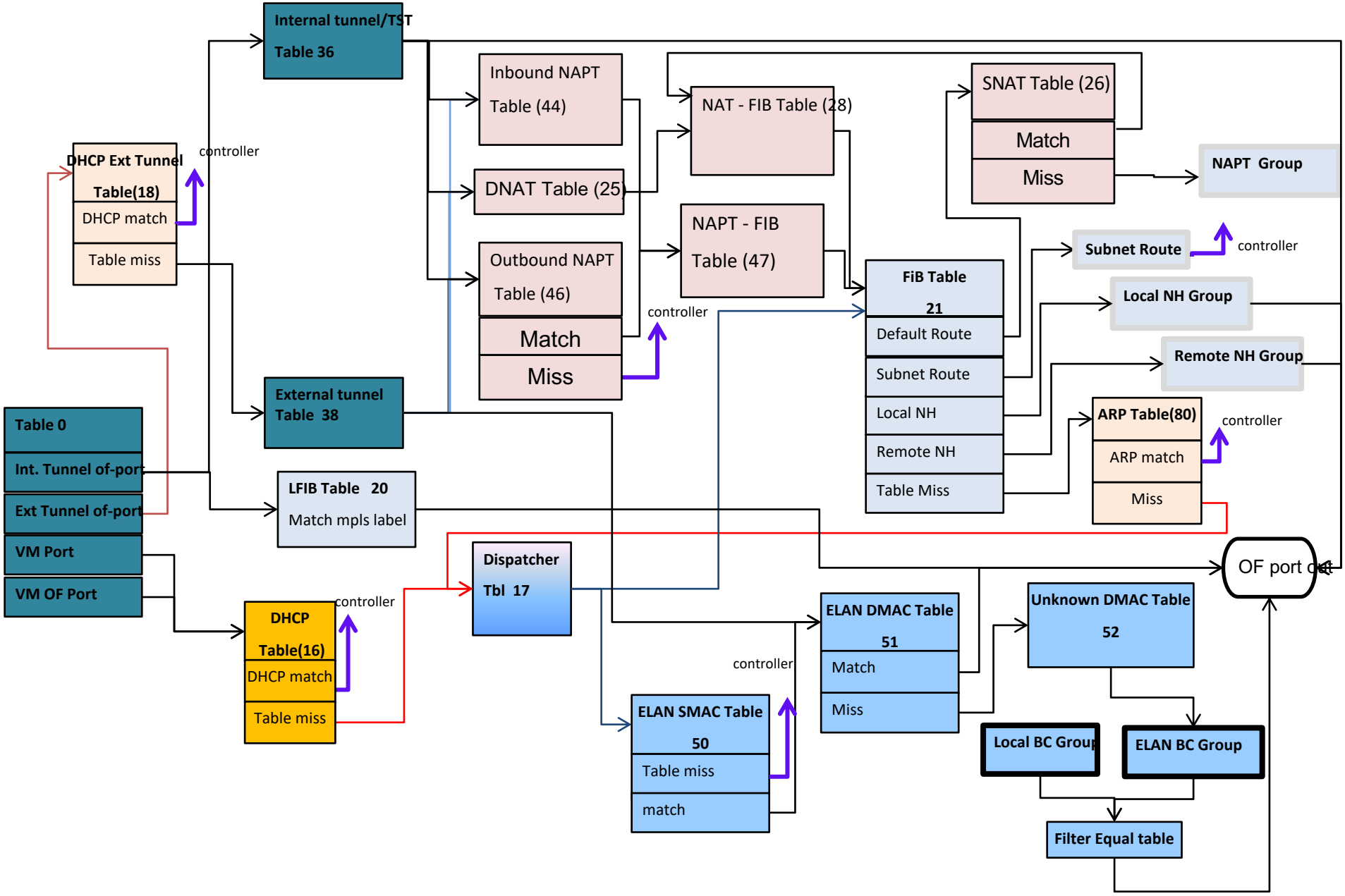
Netvirt VPN Service components in ODL (Bo)



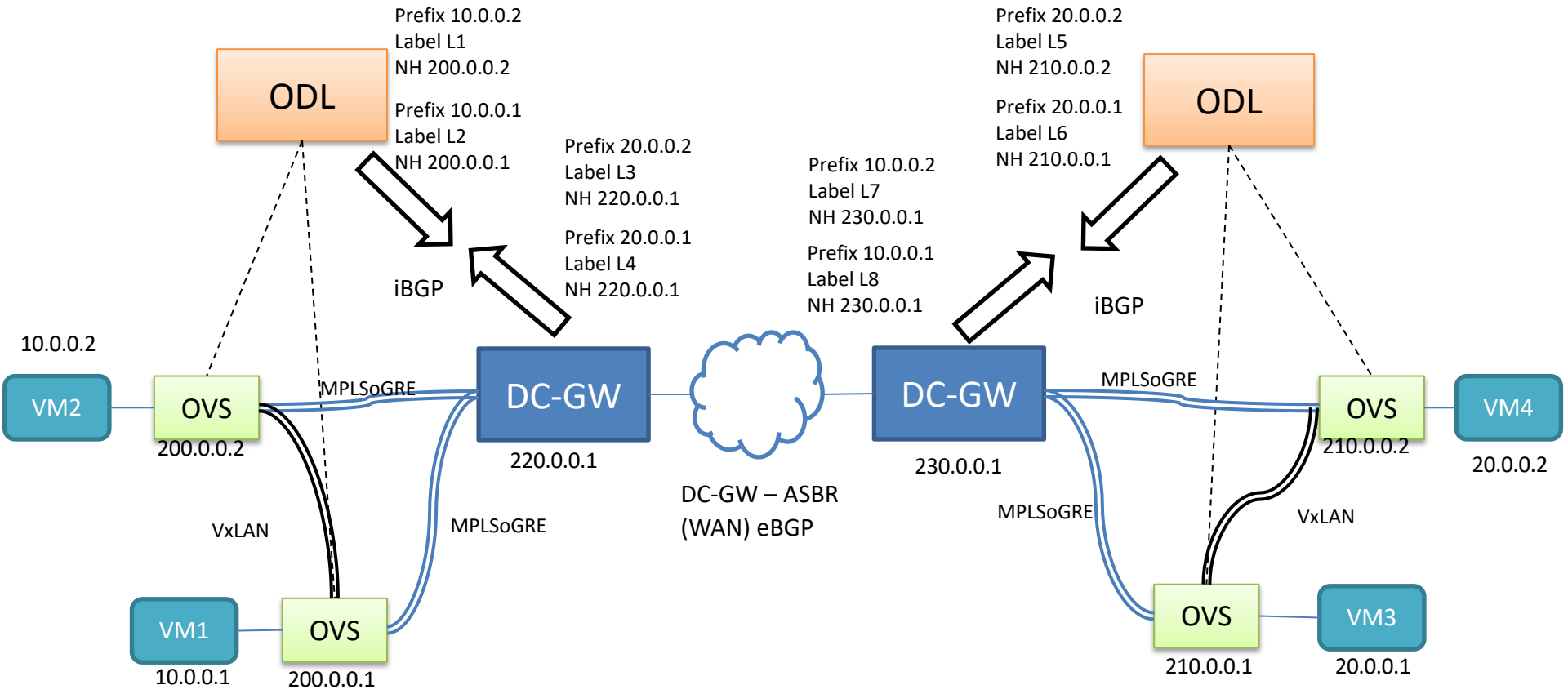
Proposed Hybrid pluggable pipeline using Genius



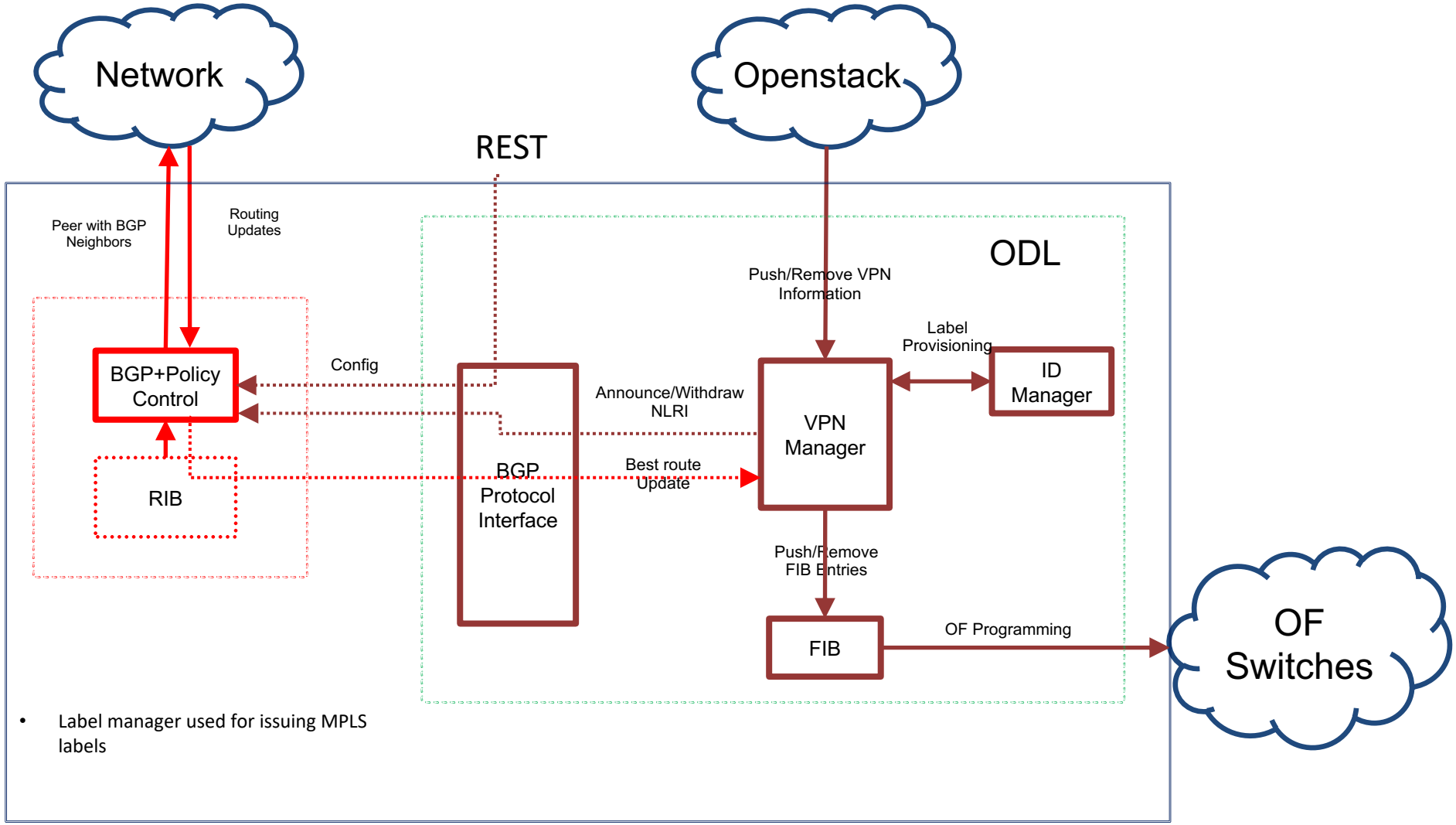
Current ODL NETVIRT VPN service pipeline



L3 VPN Architecture Overview



L3 VPN Service components



- Label manager used for issuing MPLS labels

Component Responsibilities

- VPN Manager
  - Receive s VPN instance creation/deletion requests from Openstack (via Neutron Northbound)
  - Translates VPN and Neutron router requests into VRFs, along with the set of route announce/withdraw requests
  - Uses label manager to generate labels
  - Propagates route announce/withdraw requests to the protocol manager module (currently BGP only)
  - Receives the best path route updates from the protocol manager
  - Consults the Next hop manager to resolve route nexthops into logical interfaces
  - Communicates with FIB the logical port entries for each route received

- BGP manager
  - communicates with an external BGP engine(in future LDP/OSPF) using Thrift IPC
  - Responsible for receiving best path routes from the RIB
  - Communicates the best path updates to the VPN manager
  - Communicates local routes (Prefix, NH IP, Label) from VPN Manager to the BGP engine

- FIB service
  - Receive the forwarding information from the VPN manager
  - Identify the logical output port for the route for each next hop received from VPN service (local and remote)
  - Generate per-prefix MPLS labels using ID Manager (GENIUS) , IT manager persists a mapping between (VRF ID, Prefix) and label
  - Install L3 forwarding rules (VRF, Prefix, logical output port) in FIB table for intra-DC forwarding
  - Install L3 forwarding rules (VRF, Prefix, Label, Logical output port) in LFIB table for inter-DC forwarding
  - Responds to Topology change events from Inventory Manager by identifying the set of failed route entries and reporting them to VPN manager, which will withdraw corresponding routes

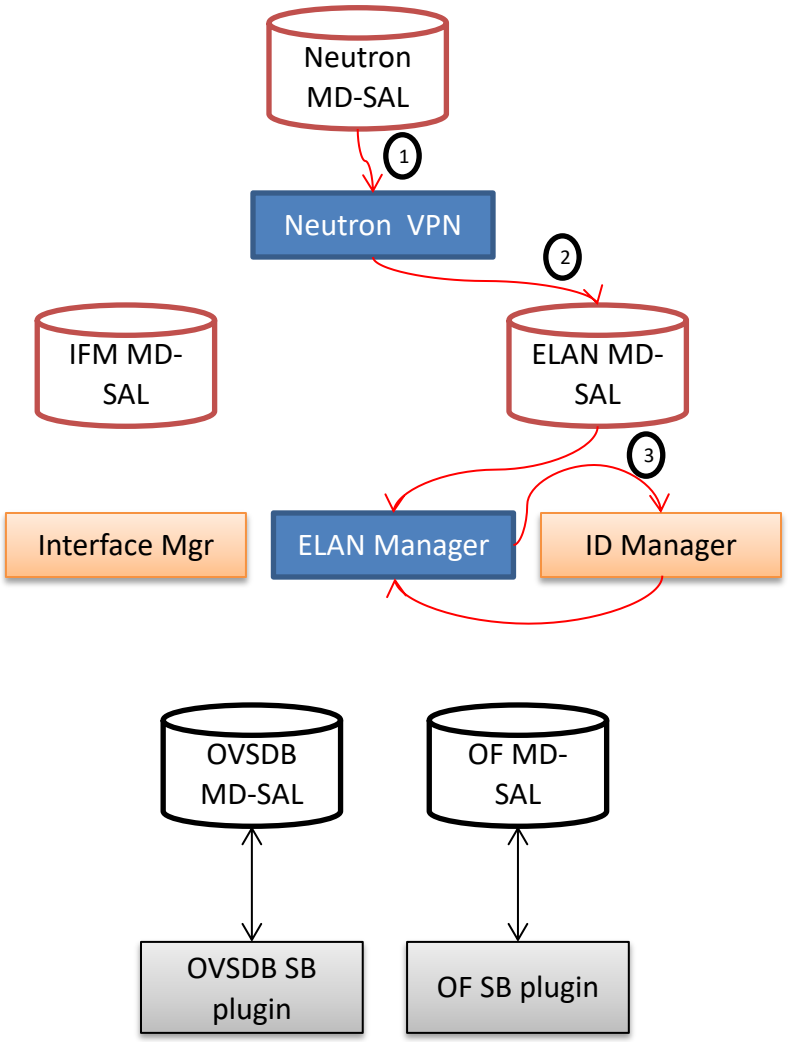
L3VPN –Next steps

- Migrate to new OF plugin
- Performance related enhancements (multi-threading, FRM bypass)
- De-coupling between FIB and VPN Manager, and migration of VPN Manager, BGP Manager back into VPN project repo
- To be discussed –
  - Continue use of service IDs for forwarding
  - Scaling considerations

L2 Service (ELAN)

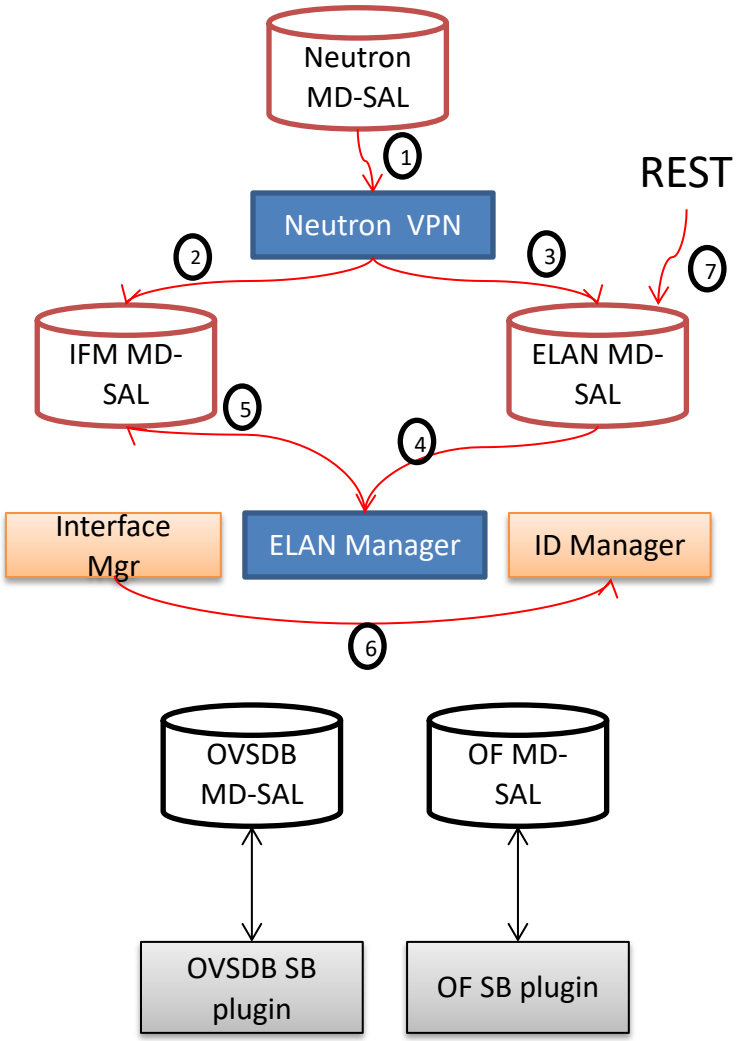
- Pt-to-pt and pt-to-multipt L2 forwarding service (MAC lookup only)
- Neutron networks map to ELAN instances and determine corresponding broadcast domains
- Static MAC support using REST API
- Support for vlan trunk ports, vlan subports, and vlan transparent networks
  - split horizon logic between subports on the same trunk port handled in the ELAN pipeline
- Forwarding between switches using VxLAN overlay
- Each ELAN maintains Local and remote broadcast groups per switch, for flooding
- MAC learning in SMAC table with punt to controller actions for miss entries
  - MAC movement supported for static and dynamically learnt MACs, based on interface manager events
  - ELAN service reactively performs flow re-writes on source and dest switches
- DMAC lookup on ingress switch only (no separate lookup on egress switch)
- Has lower priority than L3 service in GENIUS based pipeline
- Can forward non-IP pkts, IPv6 pkts (needs testing) because logic is purely L2 based

Neutron Network Create Workflow



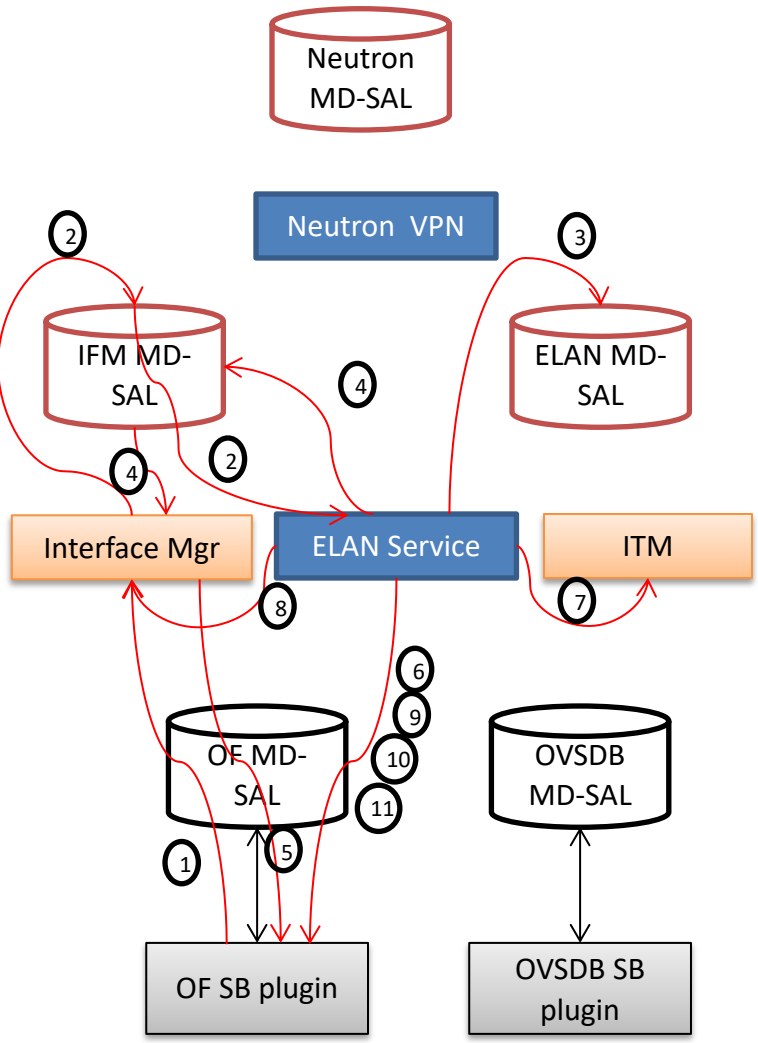
1. Neutron VPN service listens on DCNs from Neutron Northbound for network creation
2. Neutron VPN service create a new ELAN instance in ELAN DS with a new ELAN instance
3. ELAN service is activated, it allocates an ELAN tag for this instance from ID manager. The ELAN tag is a dataplane identifier for the ELAN instance. ELAN tag is persisted by ID manager across reboots.

Neutron Port Create Workflow



1. Neutron VPN service listens on DCNs from Neutron Northbound for port creation
2. Neutron VPN service create a new interface in Interface Manager with Neutron UUID as the interface name
   - The interface can be of type trunk or access, depending upon the Neutron port property. Trunk interfaces are created with Vlan ID 0.
3. Neutron VPN service creates an ELAN port in ELAN config DS for this interface
4. ELAN service receives a DCN for port creation
5. ELAN registers for interface state DCNs with Interface Manager
6. Interface manager allocates an lport tag for the interface from ID manager. The lport tag is the dataplane representation for the interface. Lport tag is persisted by ID manager across reboots.
7. Users can add static MACs on individual interfaces using ELAN service REST APIs. These entries are stored per ELAN instance, per interface in the ELAN config DS.

Neutron Port UP Workflow



1. Interface manager updates the parent ref for an interface with the OF port and DPN ID when an OF PORT UP event is received from the OF plugin
2. Interface manager fires an interface UP event, and ELAN service receives a DCN for the same, with the DPN ID for the interface
3. ELAN service maintains a list of all DPNs per ELAN instance in operDS
4. ELAN instance does a 'bind service' on the interface.
5. Interface manager adds the service into the dispatcher table (currently 17), for the 1st such bind service per switch
6. ELAN service populates the SMAC table (currently 50) on the DPN with the MAC entry for this interface. Static MAC entries are permanent, Dynamic entries are programmed to age out
7. ELAN service invokes ITM service RPC API to get the tunnel interface name to reach a destination DPN from a source DPN
8. ELAN service invokes Interface Manager RPC to get egress actions to send out packets on the tunnel interface. Interface manager will return the OF Port number for VxLAN tunnel endpoints.
9. ELAN service populates the DMAC table (currently 51) on this DPN as well as all other DPNs for the instance with the MAC address for this interface, and the corresponding egress action from Interface manager. Static MAC entries are permanent, dynamic entries are programmed to age out
10. For the 1st interface on a DPN, ELAN service creates –
    1. unknown DMAC table entry per instance per new switch associated with the instance
    2. Terminating service table
    3. Creation of ELAN broadcast group on the switch
    4. Creation of Local broadcast group on the switch
11. ELAN populates the group memberships, as well as entries in the Filter_Equals, SMAC and DMAC tables, terminating service table for each new interface on all relevant DPNs

Dataplane highlights

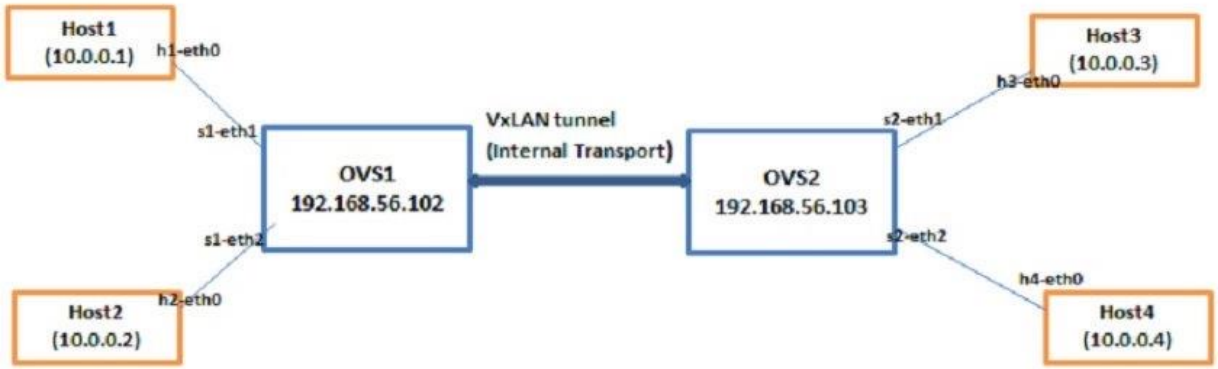
- ELAN tags and lport tags used in the pipeline, these are persisted by ID Manager across controller reboots
  - ELAN tag – 1 identifier per ELAN instance
  - Lport tag – 1 identifier per interface
- ELAN service metadata usage –
  - 3 bits of service Identifier – pushed by Interface Manager in Dispatcher table
  - 21 bits for lport tag – pushed by Interface Manager in Table 0 for ingress interface
  - 16 bits for Elan tag – pushed by ELAN service in unknown DMAC table
- SMAC table misses send a copy of the packet to the controller for learning
- DMAC table lookup sets fowarding actions as follows -
  - Unicast forwarding actions
    - Local DMAC entries have direct output port actions
    - For remote entries, DMAC flows point to the egress action (OF port) received from Interface Manager for ITM tunnel interfaces, and the lport tag is pushed in the tunnel_id field
    - Terminating service table on the egress switch does a lookup on the lport tag, and sends pkt out to the corresponding VM port (no MAC lookup)
  - Broadcast forwarding actions
    - DMAC table sends the pkt to the ELAN broadcast group, which points to the local broadcast group (group chaining), as well as remote tunnel endpoints
    - The local group forwards the pkt to all local VM ports in the same network(including VLAN subports)
    - Additionally, a pkt copy is sent to all remote TEP interfaces on the same ELAN (Head end Replication)
    - Tunnel_Id field contains the ELAN tag
    - Terminating service table on the egress switch looks up on the ELAN tag, which points to the local broadcast group, and floods the pkt to all local VM ports
  - Filter Equals table – For pkt forwarding between subports, if source and dest lport is the same, drops the packet, and if different forwards the packet to the output port

Component Classes

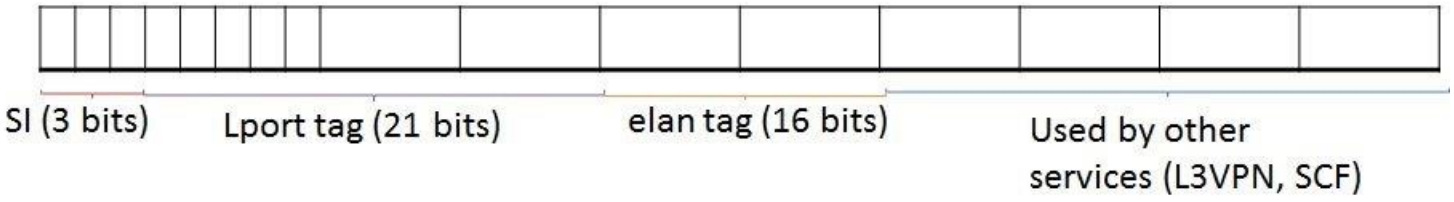
- ELAN instance manager
  - CRUD operations on ELAN instances (tied to Neutron subnet CRUD)
- ELAN interface manager
  - Service binding logic to bind ELAN instances to VM port interfaces
  - Static MAC address CRUD per ELAN instance
  - SB flow and group programming
- Interface state handler
  - SB state programming and cleanups, based on interface state events
- ELAN MAC Handler
  - Static MAC address SB programming (permanent entries), and operational DB maintenance
- Pkt-in Handler
  - Dynamic MAC learning, and corresponding operational DB maintenance
  - SB dynamic MAC address flow programming (with idle-timeouts)
- Flow Expiry Handler
  - Cleaning up of DMAC entries upon SMAC expiry, corresponding operational DB maintenance

DEMO, code walk-through

ELAN Topology



Metadata Usage



OVS1:

```
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x8000001, duration=18434.843s, table=0, n_packets=5666, n_bytes=419554, priority=5,in_port=3
actions=write_metadata:0x80000000001/0x1fffff0000000001,goto_table:36
 cookie=0x8000000, duration=18363.824s, table=0, n_packets=0, n_bytes=0, priority=4,in_port=2
actions=write_metadata:0x90000000000/0xffffff0000000000,goto_table:16
 cookie=0x8000000, duration=18361.867s, table=0, n_packets=135, n_bytes=12670, priority=4,in_port=1
actions=write_metadata:0xa0000000000/0xffffff0000000000,goto_table:16
 cookie=0x6800000, duration=19376.354s, table=16, n_packets=135, n_bytes=12670, priority=0 actions=goto_table:17
 cookie=0x8040000, duration=18343.849s, table=17, n_packets=0, n_bytes=0, priority=3,metadata=0x90000000000/0xffffff0000000000
actions=write_metadata:0x8000091388000000,goto_table:50
 cookie=0x8040000, duration=18340.819s, table=17, n_packets=135, n_bytes=12670, priority=3,metadata=0xa0000000000/0xffffff0000000000
actions=write_metadata:0x80000a1388000000,goto_table:50
 cookie=0x6800000, duration=19376.354s, table=18, n_packets=0, n_bytes=0, priority=0 actions=goto_table:38
 cookie=0x1030000, duration=19376.354s, table=20, n_packets=0, n_bytes=0, priority=0 actions=goto_table:80
 cookie=0x8000003, duration=19376.354s, table=21, n_packets=0, n_bytes=0, priority=0 actions=goto_table:80
 cookie=0x8000004, duration=19376.354s, table=22, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x9000000, duration=18436.905s, table=36, n_packets=5531, n_bytes=406884, priority=5,tun_id=0 actions=CONTROLLER:65535
 cookie=0x9001388, duration=18344.076s, table=36, n_packets=6, n_bytes=476, priority=5,tun_id=0x1388 actions=write_actions(group:209999)
 cookie=0x9000009, duration=18344.047s, table=36, n_packets=0, n_bytes=0, priority=5,tun_id=0x9 actions=output:2
 cookie=0x900000a, duration=18341.834s, table=36, n_packets=129, n_bytes=12194, priority=5,tun_id=0xa actions=output:1
 cookie=0x8050000, duration=19376.354s, table=50, n_packets=6, n_bytes=476, priority=0 actions=CONTROLLER:65535,goto_table:51
 cookie=0x8030000, duration=19376.354s, table=51, n_packets=6, n_bytes=476, priority=0 actions=goto_table:52
 cookie=0x8701388, duration=18344.047s, table=52, n_packets=6, n_bytes=476, priority=5,metadata=0x1388000000/0xffff000001
actions=write_actions(group:210000)
 cookie=0x8800009, duration=18344.015s, table=55, n_packets=0, n_bytes=0, priority=10,tun_id=0x9,metadata=0x90000000000/0x1fffff0000000000 actions=drop
 cookie=0x880000a, duration=18341.797s, table=55, n_packets=6, n_bytes=476, priority=10,tun_id=0xa,metadata=0xa0000000000/0x1fffff0000000000 actions=drop
 cookie=0x8800009, duration=18344.020s, table=55, n_packets=12, n_bytes=952, priority=9,tun_id=0x9 actions=output:2
 cookie=0x880000a, duration=18341.799s, table=55, n_packets=6, n_bytes=476, priority=9,tun_id=0xa actions=output:1
 cookie=0x1030000, duration=19376.354s, table=80, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)
mininet> sh ovs-ofctl -O OpenFlow13 dump-groups s1
OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):
 group_id=209999,type=all,bucket=actions=set_field:0x9->tun_id,resubmit(,55),bucket=actions=set_field:0xa->tun_id,resubmit(,55)
 group_id=210000,type=all,bucket=actions=group:209999,bucket=actions=set_field:0x1388->tun_id,output:3
mininet>
```

OVS2:

mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s2

OFPST_FLOW reply (OF1.3) (xid=0x2):

cookie=0x8000001, duration=18476.671s, table=0, n_packets=5678, n_bytes=420441, priority=5,in_port=3 actions=write_metadata:0x70000000001/0x1fffff0000000001,goto_table:36

cookie=0x8000000, duration=18400.630s, table=0, n_packets=8, n_bytes=616, priority=4,in_port=2 actions=write_metadata:0xb0000000000/0xffffff0000000000,goto_table:16

cookie=0x8000000, duration=18398.616s, table=0, n_packets=127, n_bytes=12054, priority=4,in_port=1 actions=write_metadata:0xc0000000000/0xffffff0000000000,goto_table:16

cookie=0x6800000, duration=19252.032s, table=16, n_packets=135, n_bytes=12670, priority=0 actions=goto_table:17

cookie=0x8040000, duration=18379.621s, table=17, n_packets=8, n_bytes=616, priority=3,metadata=0xb0000000000/0xffffff0000000000 actions=write_metadata:0x80000b1388000000,goto_table:50

cookie=0x8040000, duration=18377.606s, table=17, n_packets=127, n_bytes=12054, priority=3,metadata=0xc0000000000/0xffffff0000000000 actions=write_metadata:0x80000c1388000000,goto_table:50

cookie=0x6800000, duration=19252.032s, table=18, n_packets=0, n_bytes=0, priority=0 actions=goto_table:38

cookie=0x1030000, duration=19252.032s, table=20, n_packets=0, n_bytes=0, priority=0 actions=goto_table:80

cookie=0x8000003, duration=19252.032s, table=21, n_packets=0, n_bytes=0, priority=0 actions=goto_table:80

cookie=0x8000004, duration=19252.032s, table=22, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535

cookie=0x9000000, duration=18478.687s, table=36, n_packets=5543, n_bytes=407771, priority=5,tun_id=0 actions=CONTROLLER:65535

cookie=0x9001388, duration=18379.970s, table=36, n_packets=6, n_bytes=476, priority=5,tun_id=0x1388 actions=write_actions(group:209999)

cookie=0x900000b, duration=18379.962s, table=36, n_packets=5, n_bytes=378, priority=5,tun_id=0xb actions=output:2

cookie=0x900000c, duration=18378.579s, table=36, n_packets=124, n_bytes=11816, priority=5,tun_id=0xc actions=output:1

cookie=0x8050000, duration=19252.032s, table=50, n_packets=6, n_bytes=476, priority=0 actions=CONTROLLER:65535,goto_table:51

cookie=0x8030000, duration=19252.032s, table=51, n_packets=6, n_bytes=476, priority=0 actions=goto_table:52

cookie=0x8701388, duration=18379.967s, table=52, n_packets=6, n_bytes=476, priority=5,metadata=0x1388000000/0xffff000001 actions=write_actions(group:210000)

cookie=0x880000b, duration=18379.958s, table=55, n_packets=3, n_bytes=238, priority=10,tun_id=0xb,metadata=0xb0000000000/0x1fffff0000000000 actions=drop

cookie=0x880000c, duration=18378.574s, table=55, n_packets=3, n_bytes=238, priority=10,tun_id=0xc,metadata=0xc0000000000/0x1fffff0000000000 actions=drop

cookie=0x880000b, duration=18379.960s, table=55, n_packets=9, n_bytes=714, priority=9,tun_id=0xb actions=output:2

cookie=0x880000c, duration=18378.576s, table=55, n_packets=9, n_bytes=714, priority=9,tun_id=0xc actions=output:1

cookie=0x1030000, duration=19252.032s, table=80, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)

mininet> sh ovs-ofctl -O OpenFlow13 dump-groups s2

OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):

group_id=209999,type=all,bucket=actions=set_field:0xc->tun_id,resubmit(,55),bucket=actions=set_field:0xb->tun_id,resubmit(,55)

group_id=210000,type=all,bucket=actions=group:209999,bucket=actions=set_field:0x1388->tun_id,output:3

karaf output:

```
opendaylight-user@root>vlan:show
Name                Dpn          PortName      Vlan-Id   Tag              PortNo          AdmState      OpStateDescription
------------------------------------------------------------------------------------------
s1-eth1-trunk       1            s1-eth1       0           10             1               ENABLED       UP      null
s2-eth2-trunk       2            s2-eth2       0           11             2               ENABLED       UP      null
 s2-eth1-trunk      2            s2-eth1       0           12             1               ENABLED       UP       null
 s1-eth2-trunk      1             s1-eth2      0            9             2               ENABLED       UP      null

opendaylight-user@root>elan:show
Elan Instance                Mac-TimeOut       Tag
------------------------------------------------------------------------------------------
elan0                        30                5000
opendaylight-user@root>elaninterface:show
ElanInstance/Tag             ElanInterface/Tag       OpState
------------------------------------------------------------------------------------------
elan0/5000                   s1-eth2-trunk/9         UP
elan0/5000                   s2-eth2-trunk/11        UP
elan0/5000                   s2-eth1-trunk/12        UP
elan0/5000                   s1-eth1-trunk/10        UP

opendaylight-user@root>vxlan:show
Name                Description                 Local IP            Remote IP           Gateway IP      AdmState  OpState          Parent            Tag
------------------------------------------------------------------------------------------
TUNNEL:1            VXLAN Trunk Interface  192.168.56.102       192.168.56.103       0.0.0.0          ENABLED     UP              1/TUNNEL:1          8

TUNNEL:2            VXLAN Trunk Interface 192.168.56.103       192.168.56.102       0.0.0.0         ENABLED        UP              2/TUNNEL:2          7
```

L2 –Next steps

- Migrate to new OF plugin
- Performance related enhancements (multi-threading, FRM bypass)
- To be discussed –
  - Pros/cons of of ELAN tags and lport tags in VxLAN encap vs standard VxLAN header

Other Services – Next steps

- Security groups – Create an ACL service, and port legacy Netvirt security groups code into a stateful model – WIP
  - Also enhance the ACL service to forward pkts to SFC
- ARP – Create an ARP service, need flexibility per interface to perform ARP proxy, and learning logic with controller based learn/flood
- DHCP – create a service, allow for controller based, or openstack agent or external DHCP server to operate on a per subnet basis
- LBaaS – create service framework
- IPv6 – convert control plane functions into service. Integrate forwarding with L2 and L3 services
- QoS – service needed ?

Architectural updates needed

- Device renderers to support different types of devices as well as different versions of OVS switches
  - Each Netvirt service will consume the renderer framework
  - Can it be a common component shared with GENIUS as well for interface manager consumption
- br-int creation by controller– to be discussed
- Need each service to support DPDK and non-DPDK OVS
  - Can the test framework be run against a mix of kernel and DPDK OVS
    - Private build of DPDK OVS with conn-track available, can this be integrated into the Netvirt test environment