Proof of Concept (POC) for novel OpenStack Neutron Agent which handles both overlay within / across subnet routing and L3 routing for overlay network to external (underlay / virtual) network by leveraging use of OpenFlow Protocol and Open vSwitch

Piyush Raman,
Intern, Cloud Networking, GTS, IBM India Pvt. Ltd

INTRODUCTION

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center, all managed through a dashboard that gives administrators control and allowing users to provision resources through a web interface. OpenStack Neutron is the Networking component of OpenStack. Neutron provides “Network-as-a-Service” to interface devices (e.g. vNIC’s representing a Virtual instance) managed by OpenStack Nova.

The POC provides a novel architecture for OpenStack Neutron bridge setup. It develops an OpenStack Neutron agent for OpenStack HAVANA for ML2 Plugin which handles overlay within / across subnet routing and L3 routing for overlay network to external (underlay) network i.e. Network Address Translations (NAT) completely using flows defined by OpenFlow Protocol and thus, presents a new method to virtualize Neutron Router using SDN. We assume that the reader has basic understanding of Computer Networks and prior experience with OpenStack, OpenStack Neutron Networking Concepts including ARP Responder / L2 Population. For brief overview on ARP Responder and L2 Population refer to the Presentation accompanying the documentation

BACKGROUND

The following figure depicts the existing OpenStack Neutron setup consisting of 3 OVS Bridges- integration (BR-INT), tunnel (BR-TUN) and external (BR-EX)

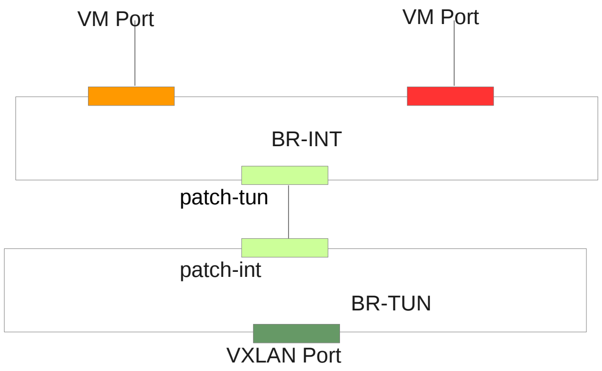
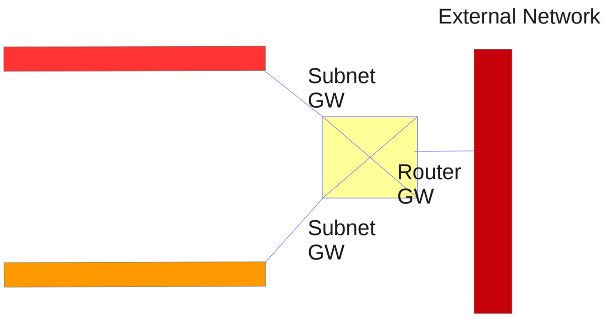


Fig 1- Bridge setup on Compute Nodes



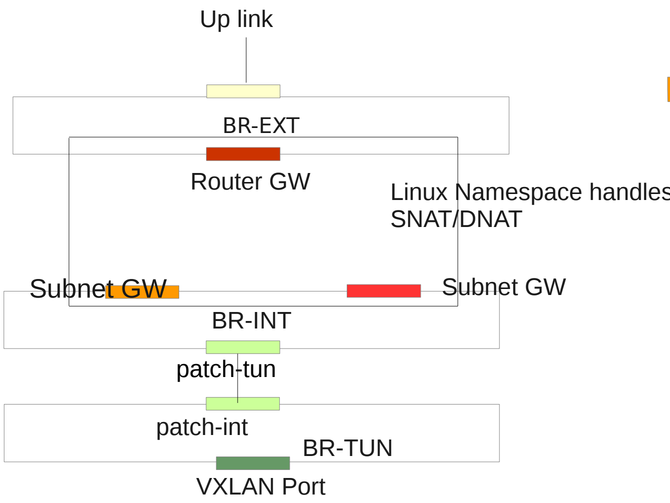


Fig2- Bridge setup on Network Node

The OVS Integration bridge acts as a normal L2 Switch (i.e. single flow having NORMAL action for all packets) and relies on the Open vSwitch Database (OVSDB) for VLAN Tagging / Untagging and packet forwarding.

OVS Tunnel contains flows defined via OpenFlow protocol by the neutron-openvswitch-agent for handling packet transmission across hosts and the local VLAN – VXLAN translations (for VXLAN Tunneling).

A Neutron Router is represented by a Linux Namespace consisting of router interfaces- subnet gateway, router gateway ports, ARP Cache and routing table associated with router. Neutron-l3-agent is responsible for creation of router ports and namespaces. It is also responsible for adding iptables rules for SNAT/DNAT. Additionally, Network node needs to enable ip forwarding to allow movement of packets across router ports in the linux namespace.

OVS external bridge acts as a normal L2 Switch (i.e. single flow having NORMAL action for all packets), having one up-link mapped to it for external network connectivity.

PROBLEM DEFINITION

Overlay across subnet and L3 routing / NATing for the traffic generated by the VM's in OpenStack is currently handled completely by the Host using Linux Namespaces, iptables and Host TCP/IP Stack, creating an additional load on the Host machine. In the existing setup (OpenStack HAVANA), each neutron-l3-agent manages one external network and each external network corresponds to only one OVS external bridge. Thus, for each external network, administrators need to launch a different instance of neutron-l3-agent and new OVS external bridge. (OpenStack ICEHOUSE introduced provider network mechanism wherein one single neutron-l3-agent can handle multiple external networks but still one external network corresponds to one OVS external bridge)

SOLUTION OVERVIEW

As a part of the new setup, overlay within / across subnet and L3 Routing / NATing is completely handled by the flows defined via OpenFlow protocol on the OVS Integration, Tunnel and External Bridge. The POC implements one unified L2-L3 OpenStack Agent which implements the mentioned functionalities instead of 2 different agents- neutron-openvswitch-agent and neutron-l3-agent. Also, one instance of the new agent handles multiple external networks instead of existing setup of using one instance of neutron-l3-agent per external network (as of OpenStack HAVANA release). The POC provides scalability in mapping multiple external network up-links on a common OVS bridge.

OpenStack Neutron router is virtualized as a Distributed Virtual Router having instance on each compute node, completely defined using flows which implements routing table and ARP cache (This functionality has been introduced in OpenStack JUNO but our implementation is different). NATing is handled by the flows defined on the OVS external bridge which maintain external network sessions on the bridge. L2 unicast / broadcast is also handled completely via flows which maintain record of VLAN Tags on each port on OVS integration Port and handle packet forwarding, thus, removing the dependency on the OVSDB for the same

The solution provides a more Software Defined Networking (SDN) approach for virtualizing Neutron Router and L3 Routing for overlay to underlay network independent of Host Stack.

SETUP DESCRIPTION

For the POC, we used VxLAN tunneling for connecting multiple compute nodes. The POC was developed for OpenStack HAVANA using ML2 Plugin for OpenStack Neutron (L2 population and ARP Responder enabled) and OVS 2.1.

The OVS Bridge setup on compute remains same but the setup on network node is modified to-

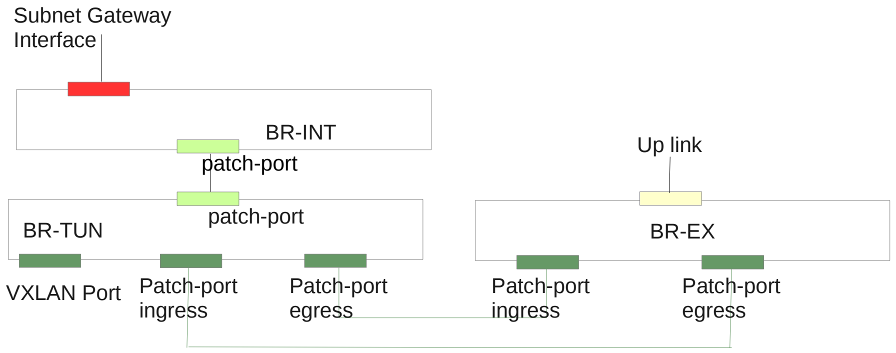


Fig 3- Network Node

For the new setup, OVS external (BR-EX) and OVS Tunnel (BR-TUN) are linked via 2 patch-ports instead of linking OVS Integration (BR-INT) and OVS external (BR-EX) using linux namespace. One patch-port handles the ingress traffic and other the egress traffic on each bridge.

POC DESCRIPTION

This section describes the flows defined on OVS integration (BR-INT) , tunnel (BR-TUN) and external (BR-EX) to achieve overlay within / across subnet routing and L3 routing for overlay to underlay (NATing).

L2 Unicast / Broadcast

Flows are defined on BR-INT to handle VLAN tagging / untagging of packets and packet forwarding (L2 unicast / broadcast) amongst ports on BR-INT based on VLAN Tags. For transmission of packet across hosts, flows are defined on BR-TUN, which handle local VLAN – global VXLAN translation for packets to-and-from VXLAN Ports. VXLAN ID for each network is identical throughout compute nodes but the VLAN tags corresponding to each network is local to each compute node and may be different on various nodes. POC uses ARP Responder mechanism, hence, flows are defined on BR-TUN to generate ARP Reply for ARP Request from ports on BR-INT on each host. L2 population mechanism is used which further limits the L2 broadcasting domain amongst VXLAN Tunnels for unknown unicasts.

Overlay across subnet

OpenStack Neutron Router is virtualized as a Distributed Virtual Router having instance on each compute node, using flows defined on BR-INT which implement Routing table / ARP Cache. Each router is identified by a Router-ID managed by the new Agent. The POC currently does not support IP packets destined to (having destination IP address) subnet gateway interfaces. Thus, any packet having destination MAC address of subnet gateway is either destined for external network or for another subnet in the overlay network. The intermediate MAC translations are handled by flows on BR-INT

L3 Routing / Network Address Translation (SNAT / DNAT) for overlay to underlay network

On each compute node, flows are defined on BR-TUN for forwarding the packets destined for external network directly to network node. On network node, flows are defined on BR-TUN for deciding if the packet belongs to overlay network or external network. It thus handles the packet forwarding to-and-from BR-EX, VXLAN ports and patch port for BR-INT unambiguously on network node. BR-TUN maintains L3 sessions for both overlay network traffic and external network which ensures that any SNAT sessions are initiated from the overlay network only. BR-EX defines flows for SNAT / DNAT and maintains L4 sessions for TCP / UDP Packets. ARP responder mechanism is also used on the BR-EX to generate ARP Replies for SNAT / DNAT Ports from the underlay network.

Following are the flow tables for various bridges-

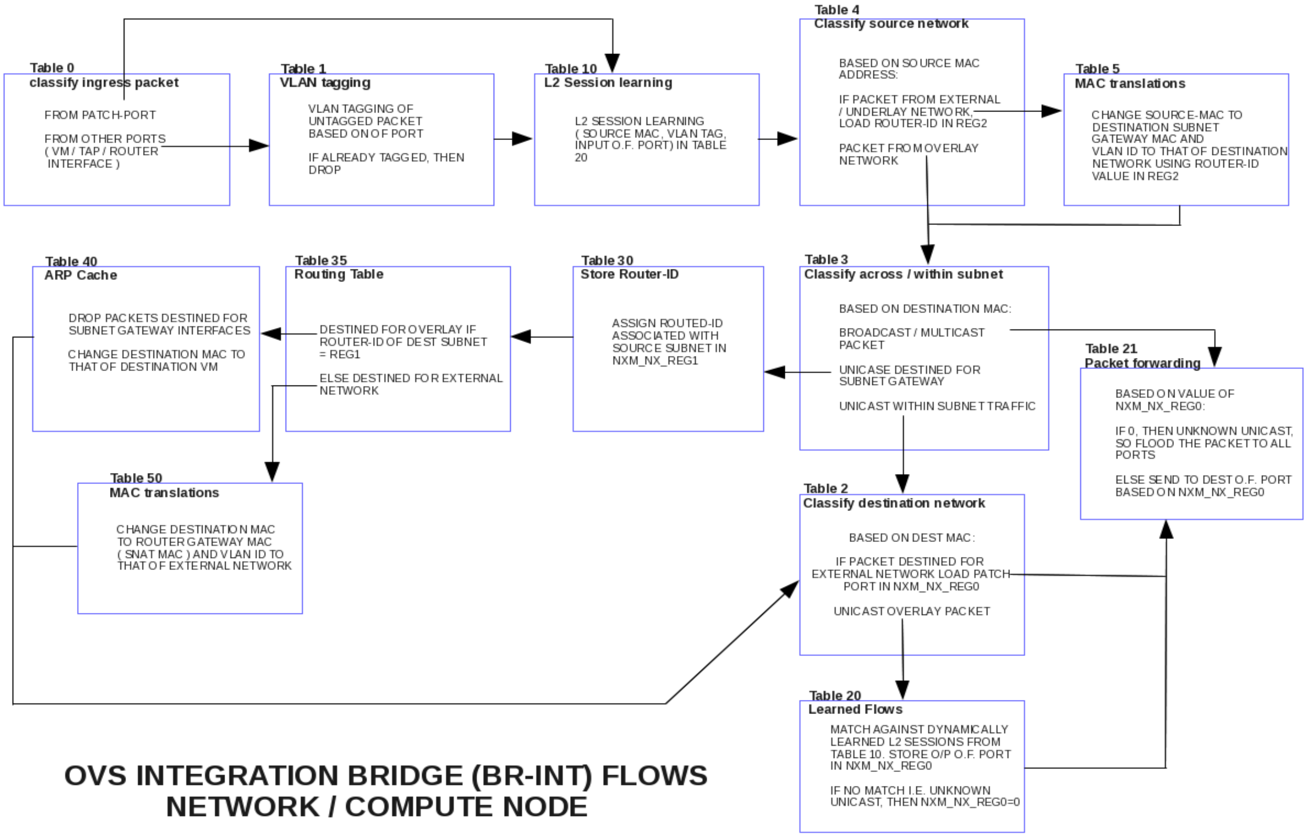


Table 0
classify ingress packet
FROM PATCH-PORT
FROM OTHER PORTS (VM / TAP / ROUTER INTERFACE)
Table 1
VLAN tagging
VLAN TAGGING OF UNTAGGED PACKET BASED ON OF PORT
IF ALREADY TAGGED, THEN DROP
Table 10
L2 Session learning
L2 SESSION LEARNING (SOURCE MAC , VLAN TAG, INPUT O.F. PORT) IN TABLE 20
Table 4
Classify source network
BASED ON SOURCE MAC ADDRESS:
IF PACKET FROM EXTERNAL / UNDERLAY NETWORK, LOAD ROUTER-ID IN REG2
PACKET FROM OVERLAY NETWORK
Table 5
MAC translations
CHANGE SOURCE-MAC TO DESTINATION SUBNET GATEWAY MAC AND VLAN ID TO THAT OF DESTINATION NETWORK USING ROUTER-ID VALUE IN REG2
Table 40
ARP Cache
DROP PACKETS DESTINED FOR SUBNET GATEWAY INTERFACES
CHANGE DESTINATION MAC TO THAT OF DESTINATION VM
Table 35
Routing Table
DESTINED FOR OVERLAY IF ROUTER-ID OF DEST SUBNET = REG1
ELSE DESTINED FOR EXTERNAL NETWORK
Table 30
Store Router-ID
ASSIGN ROUTED-ID ASSOCIATED WITH SOURCE SUBNET IN NXM_NX_REG1
Table 3
Classify across / within subnet
BASED ON DESTINATION MAC:
BROADCAST / MULTICAST PACKET
UNICASE DESTINED FOR SUBNET GATEWAY
UNICAST WITHIN SUBNET TRAFFIC
Table 21
Packet forwarding
BASED ON VALUE OF NXM_NX_REG0:
IF 0, THEN UNKNOWN UNICAST, SO FLOOD THE PACKET TO ALL PORTS
ELSE SEND TO DEST O.F. PORT BASED ON NXM_NX_REG0
Table 50
MAC translations
CHANGE DESTINATION MAC TO ROUTER GATEWAY MAC (SNAT MAC) AND VLAN ID TO THAT OF EXTERNAL NETWORK
Table 2
Classify destination network
BASED ON DEST MAC :
IF PACKET DESTINED FOR EXTERNAL NETWORK LOAD PATCH PORT IN NXM_NX_REG0
UNICAST OVERLAY PACKET
Table 20
Learned Flows
MATCH AGAINST DYNAMICALLY LEARNED L2 SESSIONS FROM TABLE 10. STORE O/P O.F. PORT IN NXM_NX_REG0
IF NO MATCH I.E. UNKNOWN UNICAST, THEN NXM_NX_REG0=0

OVS INTEGRATION BRIDGE (BR-INT) FLOWS
NETWORK / COMPUTE NODE

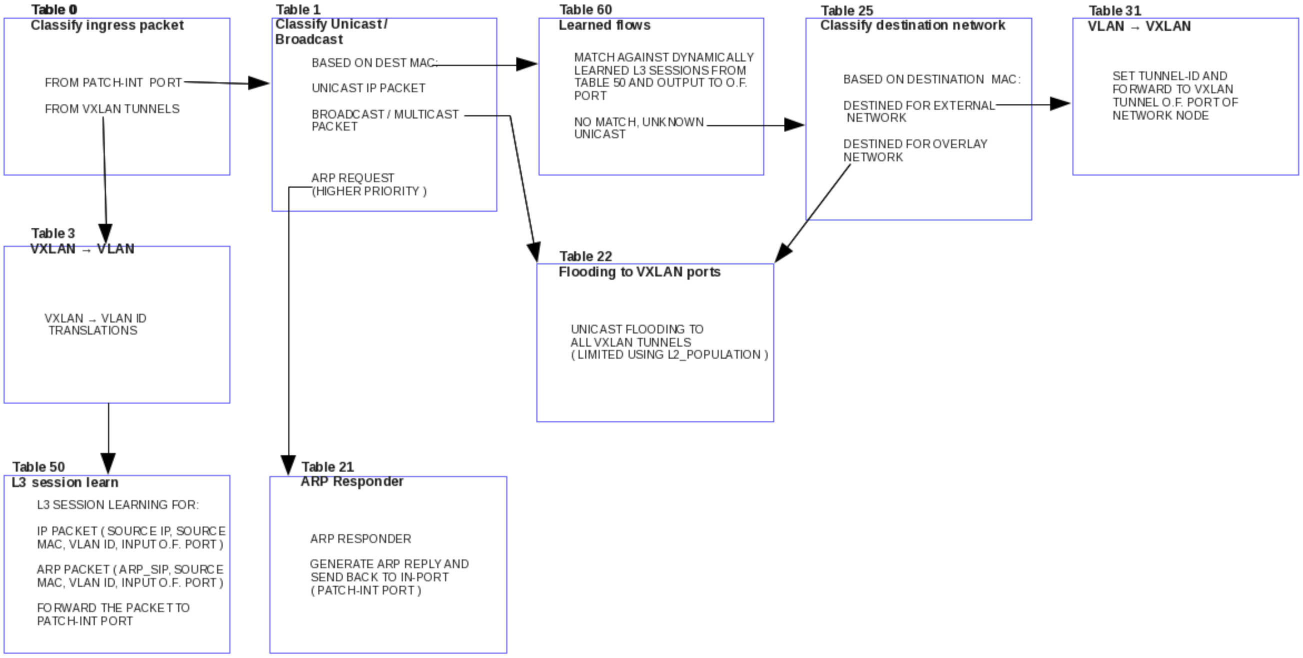


Table 0
Classify ingress packet
FROM PATCH-INT PORT
FROM VXLAN TUNNELS
Table 1
Classify Unicast / Broadcast
BASED ON DEST MAC:
UNICAST IP PACKET
BROADCAST / MULTICAST PACKET
ARP REQUEST (HIGHER PRIORITY)
Table 60
Learned flows
MATCH AGAINST DYNAMICALLY LEARNED L3 SESSIONS FROM TABLE 50 AND OUTPUT TO O.F. PORT
NO MATCH, UNKNOWN UNICAST
Table 25
Classify destination network
BASED ON DESTINATION MAC :
DESTINED FOR EXTERNAL NETWORK
DESTINED FOR OVERLAY NETWORK
Table 31
VLAN → VXLAN
SET TUNNEL-ID AND FORWARD TO VXLAN TUNNEL O.F. PORT OF NETWORK NODE
Table 3
VXLAN → VLAN
VXLAN → VLAN ID TRANSLATIONS
Table 22
Flooding to VXLAN ports
UNICAST FLOODING TO ALL VXLAN TUNNELS (LIMITED USING L2_POPULATION)
Table 50
L3 session learn
L3 SESSION LEARNING FOR:
IP PACKET (SOURCE IP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
ARP PACKET (ARP_SIP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
FORWARD THE PACKET TO PATCH-INT PORT
Table 21
ARP Responder
ARP RESPONDER
GENERATE ARP REPLY AND SEND BACK TO IN-PORT (PATCH-INT PORT)

OVS TUNNEL (BR-TUN) FLOWS
COMPUTE NODE

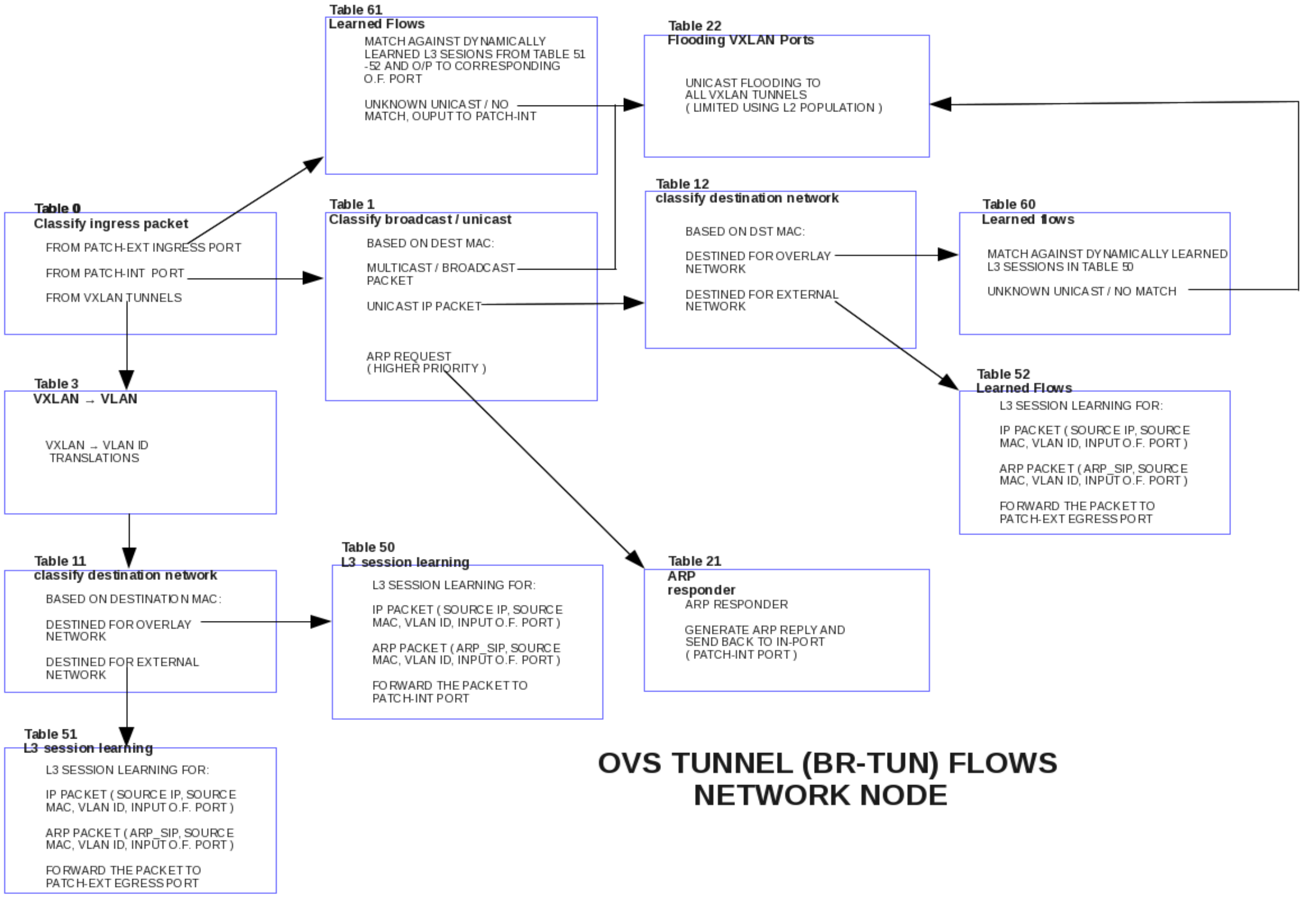


Table 0
Classify ingress packet
FROM PATCH-EXT INGRESS PORT
FROM PATCH-INT PORT
FROM VXLAN TUNNELS
Table 61
Learned Flows
MATCH AGAINST DYNAMICALLY LEARNED L3 SESIONS FROM TABLE 51 -52 AND O/P TO CORRESPONDING O.F. PORT
UNKNOWN UNICAST / NO MATCH, OUPUT TO PATCH-INT
Table 22
Flooding VXLAN Ports
UNICAST FLOODING TO ALL VXLAN TUNNELS (LIMITED USING L2 POPULATION)
Table 1
Classify broadcast / unicast
BASED ON DEST MAC:
MULTICAST / BROADCAST PACKET
UNICAST IP PACKET
ARP REQUEST (HIGHER PRIORITY)
Table 12
classify destination network
BASED ON DST MAC:
DESTINED FOR OVERLAY NETWORK
DESTINED FOR EXTERNAL NETWORK
Table 60
Learned flows
MATCH AGAINST DYNAMICALLY LEARNED L3 SESSIONS IN TABLE 50
UNKNOWN UNICAST / NO MATCH
Table 52
Learned Flows
L3 SESSION LEARNING FOR:
IP PACKET (SOURCE IP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
ARP PACKET (ARP_SIP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
FORWARD THE PACKET TO PATCH-EXT EGRESS PORT
Table 3
VXLAN → VLAN
VXLAN → VLAN ID TRANSLATIONS
Table 11
classify destination network
BASED ON DESTINATION MAC:
DESTINED FOR OVERLAY NETWORK
DESTINED FOR EXTERNAL NETWORK
Table 50
L3 session learning
L3 SESSION LEARNING FOR:
IP PACKET (SOURCE IP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
ARP PACKET (ARP_SIP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
FORWARD THE PACKET TO PATCH-INT PORT
Table 21
ARP responder
ARP RESPONDER
GENERATE ARP REPLY AND SEND BACK TO IN-PORT (PATCH-INT PORT)
Table 51
L3 session learning
L3 SESSION LEARNING FOR:
IP PACKET (SOURCE IP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
ARP PACKET (ARP_SIP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
FORWARD THE PACKET TO PATCH-EXT EGRESS PORT

OVS TUNNEL (BR-TUN) FLOWS NETWORK NODE

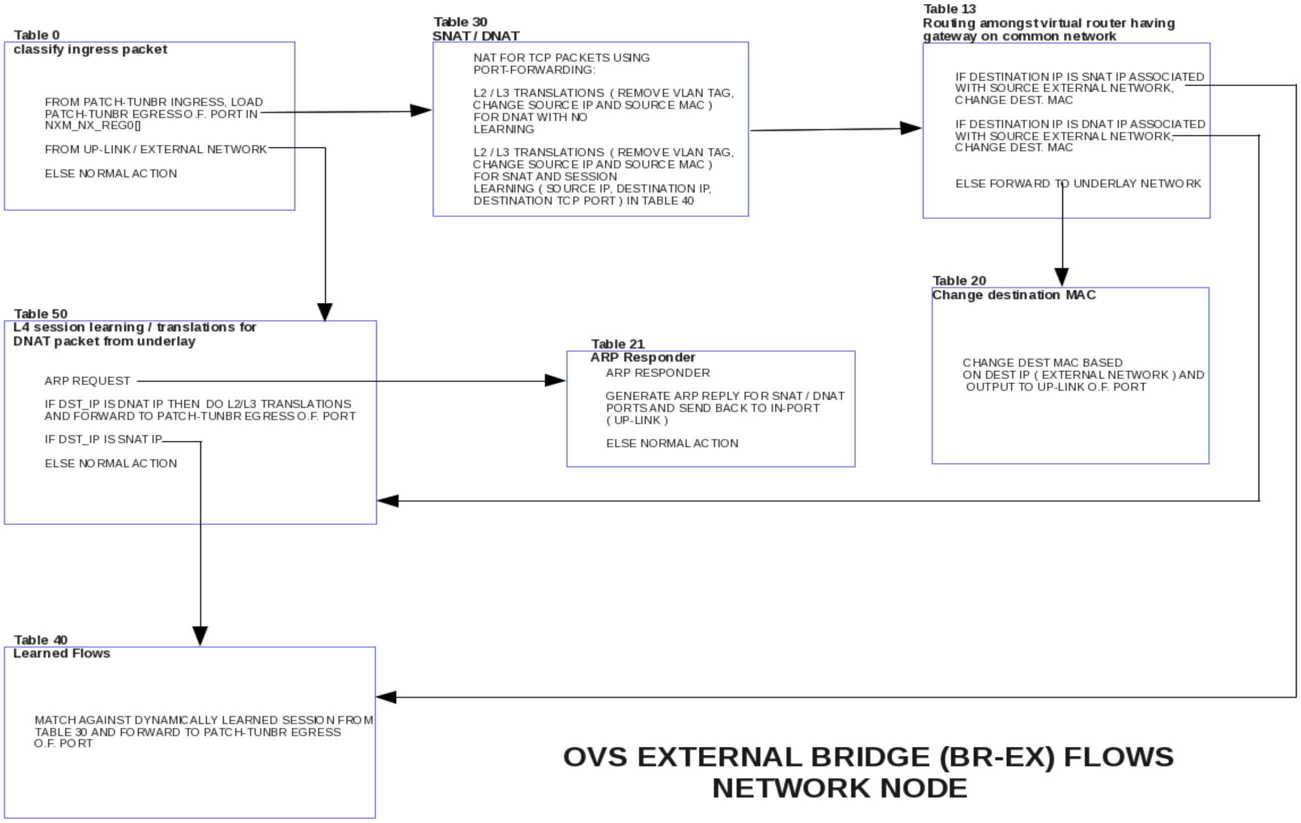


Table 0
classify ingress packet
FROM PATCH-TUNBR INGRESS, LOAD PATCH-TUNBR EGRESS O.F. PORT IN NXM_NX_REG0[]
FROM UP-LINK / EXTERNAL NETWORK
ELSE NORMAL ACTION
Table 30
SNAT / DNAT
NAT FOR TCP PACKETS USING PORT-FORWARDING:
L2 / L3 TRANSLATIONS (REMOVE VLAN TAG, CHANGE SOURCE IP AND SOURCE MAC) FOR DNAT WITH NO LEARNING
L2 / L3 TRANSLATIONS (REMOVE VLAN TAG, CHANGE SOURCE IP AND SOURCE MAC) FOR SNAT AND SESSION LEARNING (SOURCE IP, DESTINATION IP, DESTINATION TCP PORT) IN TABLE 40
Table 13
Routing amongst virtual router having gateway on common network
IF DESTINATION IP IS SNAT IP ASSOCIATED WITH SOURCE EXTERNAL NETWORK, CHANGE DEST. MAC
IF DESTINATION IP IS DNAT IP ASSOCIATED WITH SOURCE EXTERNAL NETWORK, CHANGE DEST. MAC
ELSE FORWARD TO UNDERLAY NETWORK
Table 20
Change destination MAC
CHANGE DEST MAC BASED ON DEST IP (EXTERNAL NETWORK) AND OUTPUT TO UP-LINK O.F. PORT
Table 50
L4 session learning / translations for DNAT packet from underlay
ARP REQUEST
IF DST_IP IS DNAT IP THEN DO L2/L3 TRANSLATIONS AND FORWARD TO PATCH-TUNBR EGRESS O.F. PORT
IF DST_IP IS SNAT IP
ELSE NORMAL ACTION
Table 21
ARP Responder
ARP RESPONDER
GENERATE ARP REPLY FOR SNAT / DNAT PORTS AND SEND BACK TO IN-PORT (UP-LINK)
ELSE NORMAL ACTION
Table 40
Learned Flows
MATCH AGAINST DYNAMICALLY LEARNED SESSION FROM TABLE 30 AND FORWARD TO PATCH-TUNBR EGRESS O.F. PORT

OVS EXTERNAL BRIDGE (BR-EX) FLOWS NETWORK NODE

FLOW TABLES DESCRIPTION

This section provides an overview on the functionality implemented by each table.

OVS Integration Bridge (BR-INT)
The flows on integration bridge are same for compute / network node

Table 0
It handles packet classification on the basis of ingress OpenFlow ports (VM port or patch-port) and thus decides the OpenFlow pipeline to be followed accordingly.

NOTE:
The POC currently does not handle DHCP, hence no consideration for ‘tap’ ports for DHCP

Table 1
This table handles the VLAN Tagging on untagged packets from VM ports. If the packets are already tagged, it drops them since the packets sent / received by VM’s are untagged

Table 10
This is the learning tables, which contains a single flow for maintaining L2 session via flows on Table 20

Sample flow:

```
 cookie=0x0, duration=2545.425s, table=10, n_packets=15, n_bytes=1158, idle_age=850, priority=1
actions=learn(table=20,hard_timeout=300,priority=1,NXM_OF_VLAN_TCI[0..11],NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[],load:NXM_OF_IN_PORT[]->NXM_NX_REG0[0..15]),resubmit(,4)
```

Matching part-

- -NXM_OF_VLAN_TCI[0..11]- If the VLAN tag of packet being matched in Table 20 is same as the VLAN Tag of this ingress packet
- -NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[]- If the destination MAC address of the packet being matched in Table 20 is same as the source MAC address of this ingress packet

Action part-

- -load:NXM_OF_IN_PORT[]->NXM_NX_REG0[0..15]- Load the input OpenFlow port number of the ingress packet in NXM_NX_REG0[] register

Table 4
This tables uses source MAC address and VLAN Tag to decide whether the ingress packet is from overlay network or from the external network. If the packet is from external network, it loads the router-id in NXM_NX_REG2[]

Table 5
This tables is encountered if the ingress packet is from the external network. It-

- Changes source MAC to destination subnet gateway MAC
- Changes VLAN tag to that of the destination network using the router-id in NXM_NX_REG2[] and destination IP address

Table 3
This table uses destination MAC address and VLAN tag to classify ingress packet into- (in the order of priority)

- Broadcast / Multicast Packet (such as ARP / L3 Broadcast)
- Unicast packet having destination MAC of the subnet gateway (overlay across subnet or external network)
- Unicast packet for overlay (within subnet traffic)

NOTE:
Since we do not support IP packets destined for subnet gateway interface, thus, any packet having destination MAC as subnet gateway is destined for either- overlay across subnet or external network.

Table 30
This tables stores the router-id assigned by the agent in NXM_NX_REG1[] on the basis of source subnet CIDR and VLAN tag.

Table 35
This table implements the routing table for a router, where each router is separated using router_id value in NX_NX_REG1[]. This table decides if the packet is destined for any other subnet having gateway interface on router associated with source subnet using value of NXM_NX_REG1[] and destination subnet CIDR, else forwards it to the router gateway (external network).

For overlay across subnet, this table-

- Changes source MAC to that of the destination subnet gateway MAC
- Changes VLAN tag to that of the destination network

Table 40
This tables implements the ARP Cache for overlay across subnet traffic, thus changing the destination MAC to that of the destination machine using VLAN tag and destination IP. It also implements the rule for dropping IP packets destined for subnet gateway interfaces using destination IP address

Table 50
This tables is encountered if the packet is destined for external network. It-

- Changes destination MAC to that of the router gateway interface (SNAT port)
- Changes VLAN tag to that of the external network

Table 2
This table uses destination MAC address and VLAN Tag to decide if the packet is destined for :

- External network- If so, then store the OpenFlow port value of patch port in NXM_NX_REG0[]
- Unicast overlay network

Table 20
This tables stores the learned flows from Table 10 and loads the output OpenFlow port value in NXM_NX_REG0[]

Table 21
This tables forwards the packet to destination OpenFlow port based on value in NXM_NX_REG0[]. In case the value is 0 i.e. for unknown unicast, it does unicast flooding to all ports on OVS integration bridge

OVS Tunnel Bridge (BR-TUN) - Network Node

Table 0
It handles packet classification on the basis of ingress OpenFlow ports (OVS integration patch port, OVS external ingress patch-port, VXLAN Tunnel port) and thus decided the OpenFlow pipeline to be followed accordingly

Table 3
This tables handles VXLAN → VLAN translations for packets ingress from VXLAN Tunnel ports

Table 11
This tables uses destination MAC address and VLAN Tag to classify the destination network into following categories for packets ingress from VXLAN tunnels-

- Overlay network
- External Network (if destination MAC is router gateway MAC)

Table 51
This is the learning table for packets destined for external network ingress from the VXLAN Ports. It defined flows to maintain L3 session for external network on OVS Tunnel on Table 61

Sample flow:

```
 cookie=0x0, duration=2558.717s, table=51, n_packets=0, n_bytes=0, idle_age=2558, priority=1,ip
actions=learn(table=61,priority=1,NXM_OF_VLAN_TCI[0..11],NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[],eth_type=0x800,NXM
_OF_IP_DST[]=NXM_OF_IP_SRC[],load:0->NXM_OF_VLAN_TCI[],load:NXM_NX_TUN_ID[]-
>NXM_NX_TUN_ID[],output:NXM_OF_IN_PORT[]),output:<patch_port_for_external_bridge_egress>
```

Matching part-

-NXM_OF_VLAN_TCI[0..11]- If the VLAN tag of packet being matched in Table 61 is same as the VLAN Tag of this ingress packet
-NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[]- If the destination MAC address of the packet being matched in Table 61 is same as the source MAC address of this ingress packet
-eth_type=0x0800 (IP packet)
-NXM_OF_IP_DST[]=NXM_OF_IP_SRC[]- If the destination IP Address of the packet being matched in Table 61 is same as the source IP address of this ingress packet

Action part-

-load:0->NXM_OF_VLAN_TCI[]- Remove the VLAN Tag from the packet being matched in Table 61
-load:NXM_NX_TUN_ID[]->NXM_NX_TUN_ID[]: Load the Tunnel ID of the packet being matched in Table 61 to be same as that of this ingress packet
-output:NXM_OF_IN_PORT[]: Output the matched packet in Table 61 to the input port of this ingress packet.

Table 1
This table uses destination MAC address to classify ingress packet into (in the order of priority)-

- ARP request (highest priority)
- Broadcast / Multicast Packet (L2 / L3 broadcasts)
- Unicast packet

Table 61
This table maintains the learned flows / L3 sessions for external network from table 51 and 52 and forwards the packet to destined OpenFlow port. In case of no match (for e.g. for DNAT case, where the session can be initiated from outside), the packet is unicast flooded to all VXLAN Tunnels and patch port of OVS Integration bridge

Table 22
This tables handles the unicast flooding of packet including VLAN to VXLAN translations, in case the destination VXLAN Tunnel port is unknown (e.g. DNAT session initiated from external network)

Table 50
This is the learning table for packets destined for overlay network from the VXLAN Ports. It defined flows to maintain L3 session for external network on OVS Tunnel on Table 60

Sample flow:

```
 cookie=0x0, duration=2560.551s, table=50, n_packets=1, n_bytes=98, idle_age=861, priority=1,ip
actions=learn(table=60,hard_timeout=300,priority=1,eth_type=0x800,NXM_OF_VLAN_TCI[0..11],NXM_OF_ETH_DST[]=NXM_OF_
ETH_SRC[],NXM_OF_IP_DST[]=NXM_OF_IP_SRC[],load:0->NXM_OF_VLAN_TCI[],load:NXM_NX_TUN_ID[]-
>NXM_NX_TUN_ID[],output:NXM_OF_IN_PORT[]),output:<patch_port_for_integration_bridge>
```

Matching part-

- -NXM_OF_VLAN_TCI[0..11]- If the VLAN tag of packet being matched in Table 60 is same as the VLAN Tag of this ingress packet
- -NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[]- If the destination MAC address of the packet being matched in Table 60 is same as the source MAC address of this ingress packet
- -eth_type=0x0800 (IP packet)
- -NXM_OF_IP_DST[]=NXM_OF_IP_SRC[]- If the destination IP Address of the packet being matched in Table 60 is same as the source IP address of this ingress packet

Action part-

- -load:0->NXM_OF_VLAN_TCI[]- Remove the VLAN Tag from the packet being matched in Table 60
- -load:NXM_NX_TUN_ID[]->NXM_NX_TUN_ID[]: Load the Tunnel ID of the matched packet in Table 60 to be same as that of this ingress packet
- -output:NXM_OF_IN_PORT[]: Output the matched packet in Table 60 to the incoming port of this ingress packet.

Table 60
This table maintains the learned flows / L3 sessions for overlay network from table 50 and forwards the packet to destined OpenFlow port

Table 21
This table implements the flows for ARP Responder. It generates ARP Replies for ARP Requests generated by ports on the OVS integration Bridge local to each host

Sample flow:

```
 cookie=0x0, duration=1935.722s, table=21, n_packets=0, n_bytes=0, idle_age=1935, priority=1,arp,dl_vlan=1,arp_tpa=10.10.1.2
actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],mod_dl_src:fa:16:3e:b3:6c:31,load:0x2-
>NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]-
>NXM_OF_ARP_TPA[],load:0xfa163eb36c31->NXM_NX_ARP_SHA[],load:0xa0a0102->NXM_OF_ARP_SPA[],IN_PORT
```

The flow generates an ARP reply for an ARP request for a node having IP address 10.10.1.2 and belonging to network having VLAN 1
The ARP reply contains MAC address- fa:16:3e:b3:6c:31 and ARP_SHA - 0xfa163eb36c31 (fa:16:3e:b3:6c:31) and op code 2

Table 12
This table uses destination MAC address and VLAN tag to classify destination network into following for packets ingress from OVS integration bridge patch port-

- Overlay network
- External Network

Table 52
This is the learning table for packets destined for external network from the ports on OVS integration bridge. It defined flows to maintain

L3 session for external network on OVS Tunnel on Table 61

Sample flow:

```
 cookie=0x0, duration=2558.475s, table=52, n_packets=0, n_bytes=0, idle_age=2558, priority=1,ip
actions=learn(table=61,priority=1,NXM_OF_VLAN_TCI[0..11],NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[],eth_type=0x800,NXM
_OF_IP_DST[]=NXM_OF_IP_SRC[],output:NXM_OF_IN_PORT[]),output:<patch_port_for_external_bridge_egress>
```

Matching part-

-NXM_OF_VLAN_TCI[0..11]- If the VLAN tag of packet being matched in Table 61 is same as the VLAN Tag of this ingress packet

-NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[]- If the destination MAC address of the packet being matched in Table 61 is same as the source MAC address of this ingress packet

-eth_type=0x0800 (IP packet)

-NXM_OF_IP_DST[]=NXM_OF_IP_SRC[]- If the destination IP Address of the packet being matched in Table 61 is same as the source IP address of this ingress packet

Action part-

-output:NXM_OF_IN_PORT[]: Output the matched packet in Table 61 to the input port of this ingress packet.

OVS Tunnel Bridge (BR-TUN) - Compute Node

Table 0

It handles packet classification on the basis of ingress OpenFlow ports (OVS integration patch port, VXLAN Tunnel port) and thus decided the OpenFlow pipeline to be followed accordingly

Table 3

This tables handles VXLAN → VLAN translations for packets ingress from VXLAN Tunnel ports

Table 1

This table uses destination MAC address to classify ingress packet into (in the order of priority)-

- ARP request (highest priority)
- Broadcast / Multicast Packet (L2 / L3 Broadcasts)
- Unicast packet

Table 50

This is the learning table for packets destined for overlay / external network from the VXLAN Ports. It defined flows to maintain L3 session for external network on OVS Tunnel on Table 60

Sample flow:

```
 cookie=0x0, duration=2485.194s, table=50, n_packets=31, n_bytes=3038, idle_age=808, priority=1,ip
actions=learn(table=60,hard_timeout=300,priority=1,eth_type=0x800,NXM_OF_VLAN_TCI[0..11],NXM_OF_ETH_DST[]=NXM_OF_
ETH_SRC[],NXM_OF_IP_DST[]=NXM_OF_IP_SRC[],load:0->NXM_OF_VLAN_TCI[],load:NXM_NX_TUN_ID[]-
>NXM_NX_TUN_ID[],output:NXM_OF_IN_PORT[]),output:<patch_port_for_integration_bridge>
```

Matching part-

-NXM_OF_VLAN_TCI[0..11]- If the VLAN tag of packet being matched in table 60 is same as the VLAN Tag of this ingress packet

-NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[]- If the destination MAC address of the packet being matched in table 60 is same as the source MAC address of this ingress packet

-eth_type=0x0800 (IP packet)

-NXM_OF_IP_DST[]=NXM_OF_IP_SRC[]- If the destination IP Address of the packet being matched in table 60 is same as the source IP address of this ingress packet

Action part-

-load:0->NXM_OF_VLAN_TCI[]- Remove the VLAN Tag from the packet being matched in table 60

-load:NXM_NX_TUN_ID[]->NXM_NX_TUN_ID[]: Load the Tunnel ID of the matched packet in table 60 to be same as that of this ingress packet

-output:NXM_OF_IN_PORT[]: Output the matched packet in table 60 to the incoming port of this ingress packet.

Table 22

This tables handles the unicast flooding of packet in case the destination VXLAN Tunnel port is unknown

Table 21

This table implements the flows for ARP Responder. It generates ARP Replies for ARP Requests generated by ports on the OVS

integration Bridge on the host
(Same as Table 21 for OVS Tunnel Flows on Network Node)

Table 60
This table maintains the learned flows / L3 sessions from table 50 and forwards the packet to destined OpenFlow port

Table 25
This table uses destination MAC address and VLAN Tag to decide if the destination network is-

- Overlay network
- External Network (if destination MAC is router gateway MAC)

Table 31
This tables forwards the packet destined for external network, directly to the network node

OVS External Bridge (BR-EX) - Network Node

Table 0
It handles packet classification on the basis of ingress OpenFlow ports (OVS tunnel ingress patch port or Up-link interface) and thus decided the OpenFlow pipeline to be followed accordingly

Table 50
This table classifies the ingress packet into following categories (in the order of priority)-

- ARP Request for SNAT (router gateway) / DNAT (floating-ip) interfaces
- IP packet having destination IP of DNAT interface (floating-ip)
- IP packet having destination IP of SNAT interface (router gateway)

For packet having destination IP of DNAT interface, the flows perform following translations-

- Add VLAN Tag of the external network
- Change source MAC to that of the destination subnet gateway MAC
- Change destination MAC to that of the destination VM
- Change destination IP to that of the destination VM

Table 21
This tables implements flows for ARP Responder mechanism for the SNAT/DNAT ports. It generates ARP Replies for ARP request for SNAT/DNAT ports from the external (underlay) network

NOTE
In OpenStack setup, all SNAT / DNAT ports associated with a Neutron router are represented by one single port on the OVS external bridge i.e. a single port shares multiple IP addresses. Thus, ARP Responder flows reply same MAC address for all the DNAT ports and SNAT port associated with a single router.

Table 30
This tables handles SNAT / DNATing using Port forwarding and creates learning flows to maintain L4 sessions for TCP packets in Table 40.

Sample flow for SNAT:

```
cookie=0x0, duration=423.525s, table=30, n_packets=0, n_bytes=0, idle_age=423,
priority=5,tcp,vlan_tci=0x0006/0x0fff,dl_dst=fa:16:3e:b7:62:bc
actions=learn(table=40,priority=1,eth_type=0x800,nw_proto=6,NXM_OF_TCP_DST[]=NXM_OF_TCP_SRC[],NXM_OF_IP_SRC[]=N
XM_OF_IP_DST[],NXM_OF_TCP_SRC[]=NXM_OF_TCP_DST[],NXM_OF_ETH_DST[],load:NXM_OF_VLAN_TCI[0..11]-
>NXM_OF_VLAN_TCI[0..11],load:NXM_OF_IP_SRC[]->NXM_OF_IP_DST[],load:NXM_OF_ETH_SRC[]-
>NXM_OF_ETH_DST[],load:NXM_OF_ETH_DST[]-
>NXM_OF_ETH_SRC[],output:NXM_OF_IN_PORT[]),strip_vlan,mod_nw_src:9.121.62.77,mod_dl_src:fa:16:3e:b7:62:bc,resubmit(,20
)
```

The above flow represents a SNAT translation flow and the learning tables flow created corresponding for any packet having destination MAC address of fa:16:3e:b7:62:bc and VLAN Tag 6 are SNATed to source IP address of 9.121.62.77 and source MAC address of fa:16:3e:b7:62:bc

Matching Part-

- eth_type=0x800 (IP Packet)
- nw_proto=6 (TCP)
- NXM_OF_TCP_DST[]=NXM_OF_TCP_SRC[] - If the destination TCP port of the packet being matched in Table 40 is same as

the source TCP port of the ingress packet

- NXM_OF_TCP_SRC[]=NXM_OF_TCP_DST[] - If the source TCP port of the packet being matched in Table 40 is same as the destination TCP port of the ingress packet
- NXM_OF_IP_SRC[]=NXM_OF_IP_DST[] - If the source IP address of the packet being matched in Table 40 is same as the destination IP address of the ingress packet
- NXM_OF_ETH_DST[] - If the destination MAC address of the packet being matched in Table 40 is same as the destination MAC address of the ingress packet

Action part-

- load:NXM_OF_VLAN_TCI[0..11]->NXM_OF_VLAN_TCI[0..11] – Remove VLAN tag of the pakcet being matched
- load:NXM_OF_IP_SRC[]->NXM_OF_IP_DST[] - Change destination IP address of the packet being matched in table 40 to the source IP of the ingress packet
- load:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[] - Change destination MAC address of the packet being matched in table 40 to the source MAC address of the ingress packet
- load:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[] - Change source MAC address of the packet being matched in table 40 to the destination MAC address of the ingress packet
- output:NXM_OF_IN_PORT[] - Output the packet being matched in table 40 to the ingress packet OpenFlow port

DNAT translations include:

- Changing source MAC to that of the SNAT (router gateway) MAC
- Change source IP to DNAT (router gateway) IP

There is no learning needed for the DNAT session since this can be initiated form the external network as well

Sample flow:

```
 cookie=0x0, duration=309.672s, table=30, n_packets=0, n_bytes=0, idle_age=309,
priority=10,ip,vlan_tci=0x0005/0x0fff,dl_dst=fa:16:3e:bc:0a:f3,nw_src=10.10.2.3
actions=mod_nw_src:9.121.62.74,mod_dl_src:fa:16:3e:bc:0a:f3,resubmit(,13)
```

The above flows shows DNAT for any packet having source IP of 10.10.2.3 and VLAN tag 5. The packet is DNATed to destination IP 9.121.62.74 and source MAC of fa:16:3e:bc:0a:f3 and resubmitted to Table 13

(As mentioned earlier, the source MAC for both SNAT/DNAT is same (router gateway MAC))

Table 13
This table handles routing amongst VM's in overlay network connected via different virtual routers (using SNAT / DNAT IP). For e.g. a VM in the overlay initiates an IP session for floatingIP- so in this case the source and destination both lie in the overlay.

This tables has 2 functionalities (in the order of priority)-

- It drops any packet initiated from non floatingIP-associated VM destined for its own SNAT IP (this is the case when source and destination IP address are same after SNAT for non-floatingIP-associated VM's)
- Check for routing amongst VM's on networks connected by different virtual routers. (if the source and destination both lie in the overlay itself, for e.g.- IP session for floatingIP's). If so, this table handles destination MAC address translations for valid traffic.

If none of the flows match, then the VLAN tag is remove and packet is forwarded to the underlay external network

Table 40
This tables maintains the learned flows from table 30

Table 20
This tables implements the ARP Cache for external network (underlay network nodes). The flows for this table needs to be added manually by the user.

FUNCTIONALITIES SUPPORTED

The POC in its current form cannot maintain unique ICMP SNAT session, since we cannot currently access ICMP identifier field. ICMP SNAT session for a given destination IP can be initiated by only one VM at a time, amongst a group of non floatingIP-associated VM's sharing common SNAT IP. TCP SNAT sessions can still be maintained uniquely because the learning is based on MAC address, IP address, IP protocol and TCP ports. Note that we currenlty use Port Forwarding and not Port Mapping (The ideal way is to implement port mapping and not port forwarding)

Sample learned flow in table 40 on OVS external bridge for ICMP

table=40, hard_timeout=300, priority=1,**icmp,dl_dst=fa:16:3e:12:58:cb,nw_src=9.121.62.67** actions=load:0x5-

```
>NXM_OF_VLAN_TCI[0..11],load:0xa0a0203->NXM_OF_IP_DST[],load:0xfa163ecdbc43->NXM_OF_ETH_DST[],load:0xfa163e1258cb->NXM_OF_ETH_SRC[],output:3
```

Multiple VM's sharing common SNAT IP share common router gateway MAC so we cannot uniquely identify the ICMP SNAT sessions. ICMP SNAT session learning is based on destination MAC, source IP and protocol only

Sample learned flow in table 40 on OVS external bridge for TCP

```
table=40, hard_timeout=300, prority=1,tcp,dl_dst=fa:16:3e:12:58:cb,nw_src=9.121.62.66,tp_src=22,tp_dst=37965 actions=load:0x5->NXM_OF_VLAN_TCI[0..11],load:0xa0a0103->NXM_OF_IP_DST[],load:0xfa163ebbb36a->NXM_OF_ETH_DST[],load:0xfa163e1258cb->NXM_OF_ETH_SRC[],output:3
```

TCP SNAT session involves TCP source and destination ports additionally

For example, if we have 4 VM's sharing subnet gateway on one common router and another 4 VM's sharing subnet gateway on another router, with no floatingIP-associations, as shown below then we have 2 groups-

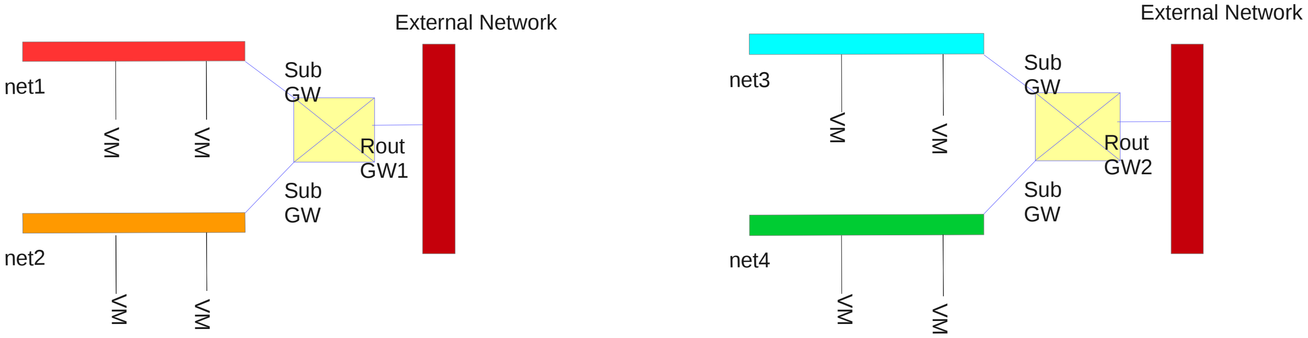


Fig- 4

Group 1- 4 VM's → inst_net1, inst_net1_2, inst_net2, inst_net2_2 (SNAT_IP1, router_gateway_MAC1)

Group 2- 4 VM's → inst_net3, inst_net3_2, inst_net4, inst_net4_2 (SNAT_IP2, router_gateway_MAC2)

Hence for each group mentioned above only one VM within a group can be used for ICMP SNATing for a given external network destination IP at a time.

Lets consider following specification for Group 1-

VM	**IP**	**MAC**
inst_net1	IP_net1	MAC_net1
inst_net1_2	IP_net1_2	MAC_net1_2
inst_net2	IP_net2	MAC_net2
inst_net2_2	IP_net2_2	MAC_net2_2

Lets consider following specification for Group 2-

VM	**IP**	**MAC**
inst_net3	IP_net3	MAC_net3
inst_net3_2	IP_net3_2	MAC_net3_2
inst_net4	IP_net4	MAC_net4
inst_net4_2	IP_net4_2	MAC_net4_2

Consider, inst_net1 starts an ICMP session for external node having IP (9.121.62.66). So on OVS external bridge, following rule is encountered in Table 30 on OVS external bridge (BR-EX) -

```
table=30,dl_type=0x0800,nw_proto=1,vlan_tci=0x0005/0x0fff,dl_dst=fa:16:3e:b7:62:bc,actions=learn(table=40,priority=1,eth_type=0x800,nw_proto=6,NXM_OF_TCP_DST[]=NXM_OF_TCP_SRC[],NXM_OF_IP_SRC[]=NXM_OF_IP_DST[],NXM_OF_TCP_SRC[]=NXM_OF_TCP_DST[],NXM_OF_ETH_DST[],load:NXM_OF_VLAN_TCI[0..11]->NXM_OF_VLAN_TCI[0..11],load:NXM_OF_IP_SRC[]->NXM_OF_IP_DST[],load:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],load:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[],output:NXM_OF_IN_PORT[]),strip_vlan,mod_nw_src:9.121.62.77,mod_dl_src:fa:16:3e:b7:62:bc,resubmit(,20)
```

where
fa:16:3e:b7:62:bc – router_gateway_MAC1
9.121.62.77- SNAT_IP1

So the packet from inst_net1 (VM) is given a SNAT IP of 9.121.62.77 with router gateway MAC of fa:16:3e:b7:62:bc and forwarded to underlay network. Also a learning flow is created for this session in Table 40 on OVS external bridge (BR-EX)

table=40, hard_timeout=300, priority=1,**icmp,dl_dst=fa:16:3e:b7:62:bc,nw_src=9.121.62.66** actions=load:0x5->NXM_OF_VLAN_TCI[0..11],load:**0xa0a0203**->NXM_OF_IP_DST[],load:**0xfa163ecdbc43**->NXM_OF_ETH_DST[],load:**0xfa163eb762bc**->NXM_OF_ETH_SRC[],output:3

where -
0xa0a0203 → IP_net1
0xfa163ecdbc43 → MAC_net1
0xfa163eb762bc → router_gateway_MAC1

Now note that in Table 40, for maintaining the session of VM inst_net1, we only use destination MAC address and source IP address of the ICMP reply coming in.

Also now lets assume if another VM- inst_net1_2 also initiates an ICMP session for external network (9.121.62.66). So table 30 on OVS external bridge (BR-EX) is encountered again-

table=30,dl_type=0x0800,nw_proto=1,vlan_tci=0x0005/0x0fff,**dl_dst=fa:16:3e:b7:62:bc**,actions=learn(table=40,priority=1,eth_type=0x800,nw_proto=6,**NXM_OF_TCP_DST[]=NXM_OF_TCP_SRC[],NXM_OF_IP_SRC[]=NXM_OF_IP_DST[],NXM_OF_TCP_SRC[]=NXM_OF_TCP_DST[],NXM_OF_ETH_DST[],load:NXM_OF_VLAN_TCI[0..11]->NXM_OF_VLAN_TCI[0..11],load:NXM_OF_IP_SRC[]->NXM_OF_IP_DST[],load:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],load:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[]**,output:NXM_OF_IN_PORT[]),strip_vlan,mod_nw_src:**9.121.62.77**,mod_dl_src:**fa:16:3e:b7:62:bc**,resubmit(,20)

where
fa:16:3e:b7:62:bc – router_gateway_MAC1
9.121.62.77- SNAT_IP1

So the packet from inst_net1_2 (VM) is given a SNAT IP of 9.121.62.77 with router gateway MAC of fa:16:3e:b7:62:bc and forwarded to underlay network. Also a learning flow is created for this session in Table 40 on OVS external bridge (BR-EX)

table=40, hard_timeout=300, priority=1,**icmp,dl_dst=fa:16:3e:b7:62:bc,nw_src=9.121.62.66** actions=load:0x5->NXM_OF_VLAN_TCI[0..11],load:**0xa0a0203**->NXM_OF_IP_DST[],load:**0xfa163ecdbc43**->NXM_OF_ETH_DST[],load:**0xfa163eb762bc**->NXM_OF_ETH_SRC[],output:3

where -
0xa0a0203 → IP_net1
0xfa163ecdbc43 → MAC_net1
0xfa163eb762bc → router_gateway_MAC1

Now note that in Table 40 on OVS external bridge (BR-EX), for maintaining the session of VM inst_net1_2, we only use destination MAC address and source IP address of the ICMP reply coming in.

So we observe that for a group of VM's sharing common SNAT IP / router gateway MAC (all the VM's in Group 1 - inst_net1, inst_net1_1, inst_net2, inst_net2_2), if they are all generating ICMP sessions for same destination IP (here 9.121.62.66) at the same time, they share common Router Gateway MAC and source IP for the learned flow session in Table 40 on OVS external bridge (BR-EX). Hence, we cannot differentiate amongst different ICMP sessions for different VM's within a group sharing common SNAT IP and router gateway MAC.

Same applies for group 2 as well

However, all such groups can be checked for the common destination IP simultaneously because they have different SNAT IP and router gateway MAC address but only one within a group.

That is if lets say 2 VM's of different grupus- inst_net1 (Group 1) and inst_net3 (Group 2) start ICM session for common destination IP (9.121.62.66), both will hit tables 30 in OVS external bridge but will hit different flows on table 30 because they have different router gateway MAC, so "dl_dst" parameter would be different for both.

Example-inst_net1 will hit-

table=30,dl_type=0x0800,nw_proto=1,vlan_tci=0x0005/0x0fff**,dl_dst=fa:16:3e:b7:62:bc**,actions=learn(table=40,priority=1,eth_type=0x800,nw_proto=6,**NXM_OF_TCP_DST[]=NXM_OF_TCP_SRC[],NXM_OF_IP_SRC[]=NXM_OF_IP_DST[],NXM_OF_TCP_SRC[]=NXM_OF_TCP_DST[],NXM_OF_ETH_DST[],load:NXM_OF_VLAN_TCI[0..11]->NXM_OF_VLAN_TCI[0..11],load:NXM_OF_IP_SRC[]->NXM_OF_IP_DST[],load:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],load:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[]**,output:NXM_OF_IN_PORT[]),strip_vlan,mod_nw_src:**9.121.62.77**,mod_dl_src:**fa:16:3e:b7:62:bc**,resubmit(,20)

where
fa:16:3e:b7:62:bc – router_gateway_MAC1
9.121.62.77- SNAT_IP1

and create a learned flow in table 40 on OVS external bridge (BR-EX)

table=40, hard_timeout=300, priority=1,**icmp,dl_dst=fa:16:3e:b7:62:bc,nw_src=9.121.62.66** actions=load:0x5->NXM_OF_VLAN_TCI[0..11],load:**0xa0a0203**->NXM_OF_IP_DST[],load:**0xfa163ecdbc43**->NXM_OF_ETH_DST[],load:**0xfa163eb762bc**->NXM_OF_ETH_SRC[],output:3

where -
0xa0a0203 → IP_net1
0xfa163ecdbc43 → MAC_net1
0xfa163eb762bc → router_gateway_MAC1

and inst_net3's packet will hit following table 30 flow on OVS external bridge (BR-EX)-

table=30,dl_type=0x0800,nw_proto=1,vlan_tci=0x0005/0x0fff,**dl_dst=fa:16:3e:ef:fa:e8**,actions=learn(table=40,hard_timeout=300,priority=1,eth_type=0x800,**NXM_OF_IP_PROTO[],NXM_OF_IP_SRC[]=NXM_OF_IP_DST[],NXM_OF_ETH_DST[],load:NXM_OF_VLAN_TCI[0..11]->NXM_OF_VLAN_TCI[0..11],load:NXM_OF_IP_SRC[]->NXM_OF_IP_DST[],load:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],load:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[],**output:NXM_NX_REG0[]),mod_nw_src:**9.121.62.70**,mod_dl_src:**fa:16:3e:ef:fa:e8**,resubmit(,20)

where
fa:16:3e:ef:fa:e8 – router_gateway_MAC2
9.121.62.70- SNAT_IP2

and create a learned flow in table 40 on OVS external bridge (BR-EX)

table=40, hard_timeout=300, priority=1,**icmp,dl_dst=fa:16:3e:ef:fa:e8,nw_src=9.121.62.66** actions=load:0x5->NXM_OF_VLAN_TCI[0..11],load:**0xa0a0103**->NXM_OF_IP_DST[],load:**0xfa163ecdab43**->NXM_OF_ETH_DST[],load:**0xfa163eeffae8**->NXM_OF_ETH_SRC[],output:3

where -
0xa0a0103 → IP_net3
0xfa163ecdab43 → MAC_net3
0xfa163eeffae8 → router_gateway_MAC2

Thus two different sessions are maintained for 2 VM's belonging to two different groups with each group having common SNAT IP and router gateway MAC.

So we can summarize, for a node in external network having IP- 9.121.62.66 or 9.121.62.67

Only one of the VM's in Group 1 can Ping 9.121.62.66 at a time because they share common router gateway MAC, and,

Only one of the VM's in Group 2 can Ping 9.121.62.66 at a time because they share common router gateway MAC, but

2 different VM's in Group 1 can ping together considering one pings 9.121.2.66 and other pings 9.121.62.67 at the same time
Same holds for Group 2, also,

A VM from Group 1 and another VM from Group 2 can both ping 9.121.62.66 at the same time since they have different router gateway MAC

There is no such hard constraint for TCP SNAT sessions. For TCP packets, Port Forwarding is used so ports collision is less likely amongst the VM's in each group since TCP session learning involves source and destination TCP ports. However, collision is still

possible if the source and destination port for any 2 VM's SNAT session in the same group happens to be same. User can initiate TCP sessions simultaneously (e.g. ssh sessions to external nodes) from VM's within a group.
To ensure that no port collision is happening for TCP packets, user can dump flows in Table 40 on OVS external bridge, which maintains SNAT TCP sessions. Thus total number of unique TCP SNAT sessions being maintained can be cross checked from these dumps in Table 40. Solution for this port collision is described in "FUTURE WORK" section

Another drawback for SNAT is that since we cannot uniquely identify the SNAT ICMP session, thus any ICMP packet destined for SNAT IP from external network may be accepted instead of being dropped and forwarded to the VM in the overlay. (SNAT sessions cannot be initiated from the outside / underlay). This may happen if lets say VM (10.10.1.2) is pinging external node (9.121.62.66) with SNATed IP 9.121.62.70. Now a learning session is created on BR-EX / OVS external bridge based on destination MAC address, source IP address and IP protocol in Table 40 for the session. Now, if the connection is terminated by the VM the learning flows still remain on the bridge and are removed after a certain time-out value (100 seconds for POC). Now if the external node (9.121.62.66) initiates an ICMP session for 9.121.62.70 (SNATed IP) before time-out, then the flows cannot determine if the session was initiated from overlay or the underlay and would forward the packet to the same overlay VM as if the session was initiated by the VM in the overlay.

The POC currently does not support IP packets destined for SNAT interfaces. Thus, users need to ensure that no sessions are initiated from outside to SNAT IP. (This problem arises since we cannot identify SNAT sessions uniquely).

NOTE:
Since floatingIP's are part of external network, all the constraint defined for external network IP also apply to floatingIP's.

For ICMP SNAT, only single VM within a group sharing common SNAT IP must ping a common floatingIP at a time. Multiple VM's within a group sharing common SNAT IP are not allowed to send ICMP packets destined for a common floating-IP (Since we cannot uniquely identify each SNAT session, as mentioned in detail earlier in detail for external networks)

For example, if we have 4 VM's sharing subnet gateway on one common router and another 4 VM's sharing subnet gateway on another router, with no floatingIP-associations, (e.g. topology in Fig 7) then we have 2 groups-

Group 1- 4 VM's → inst_net1, inst_net1_2, inst_net2, inst_net2_2 (SNAT_IP1, router_gateway_MAC1)

Group 2- 4 VM's → inst_net3, inst_net3_2, inst_net4, inst_net4_2 (SNAT_IP2, router_gateway_MAC2)

Lets consider following specification for Group 1-

VM	**IP**	**MAC**	**FloatingIP-associated**
inst_net1	IP_net1	MAC_net1	Yes
inst_net1_2	IP_net1_2	MAC_net1_2	No
inst_net2	IP_net2	MAC_net2	No
inst_net2_2	IP_net2_2	MAC_net2_2	Yes

Lets consider following specification for Group 2-

VM	**IP**	**MAC**	**FloatingIP-associated**
inst_net3	IP_net3	MAC_net3	No
inst_net3_2	IP_net3_2	MAC_net3_2	Yes
inst_net4	IP_net4	MAC_net4	Yes
inst_net4_2	IP_net4_2	MAC_net4_2	No

So for destination IP address as 9.121.62.80 (lets say floatingIP for inst_net3_2)

Only one amongst the non floatingIP-associated VM's in Group 2- inst_net3 or inst_net4_2 can Ping 9.121.62.80 at a time because they share common router gateway MAC, and,

Only one of the non floatingIP-associated VM's in Group 1- inst_net1_2 and inst_net2 can Ping 9.121.62.80 at a time because they share common router gateway MAC, but

2 different non floatingIP-associated VM's within Group 2 – inst_net3 can ping 9.121.62.80 and int_net4_2 can ping 9.121.62.77 (floatingIP associated with int67_net1) at the same time

Same holds for other Group 1, also,

2 different non floatingIP-associated VM's of different groups – inst_net2 and int_net4_2 can both ping 9.121.62.80 at the same time

Like in case of external network, POC cannot distinguish whether the SNAT IP session session was initiated by SNATed VM or not. Thus,

the user needs to take care to not initiate any session to the SNAT IP's. (as mentioned in detail earlier)
The following tables summarizes all the Use-Cases that a user can try out with the POC along with the constraints, if any

USE-CASE	**CONSTRAINTS**
Overlay across / within subnet traffic (IP / ARP)	- No constraints
IP sessions from non floatingIP-associated VM to external network / another floating-IP	- ICMP sessions for a common destination IP can be initiated by one VM at a time within a group sharing common SNAT IP - No constraints for TCP sessions until no port collision
IP sessions initiated from external network / floatingIP-associated VM to SNAT IP	- User must explicitly take care of that no IP sessions are INITIATED TO SNAT IP
IP sessions from one non floatingIP-associated VM to another using destination IP address as SNAT IP	- User must explicitly take care of that no IP sessions are INITIATED TO SNAT IP
IP sessions from floatingIP-associated VM to-and-from external netwoks (SNAT / DNAT – floatingIP)	- No constraints
IP sessions from one floatingIP-associated VM to another	- No constraints

NOTE:

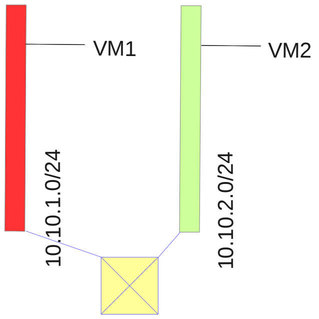
The POC currently does not programatically handle multiple uplinks / external networks on a common bridge. The same can be achieved with minimal efforts by updating the ARP cache table in OVS external bridge manually (adding flows in table 20 and in table 0 for the new up-link). However, the administrator must ensure that there is no redundancy involved amongst the multiple external network's IP pool.

ADVANTAGES OF POC

- Overlay across subnet routing using flows on OVS integration and tunnel bridge
Overlay across subnet routing including MAC / IP / VLAN Tag translations are completely handled using flows defined on OVS integration and tunnel against the existing mechanism of using Linux Namespaces and Host TCP/IP Stack for the same. Thus, Neutron Router (including routing table and ARP cache) is virtualized using flows on OVS Integration Bridge, completely eliminating the dependency of Host system. (Refer Fig. 5) Thus there is no more need of enabling IP_forwarding on any node

- Decentralize overlay across subnet routing decision on each node
Overlay across subnet routing decisions is de-centralized on each compute node against the existing mechanism of centralized routing decisions on the network node. This reduces the load and dependency on the Network Node which acts as a single point of failure for overlay across subnet routing in the existing architecture and reduces underlay traffic. (Refer Fig. 5)

NOTE-
This feature has been introduced in OpenStack Community as Distributed Virtual Routing (DVR). However, the entire mechanism for the same is different from what is being used in the POC



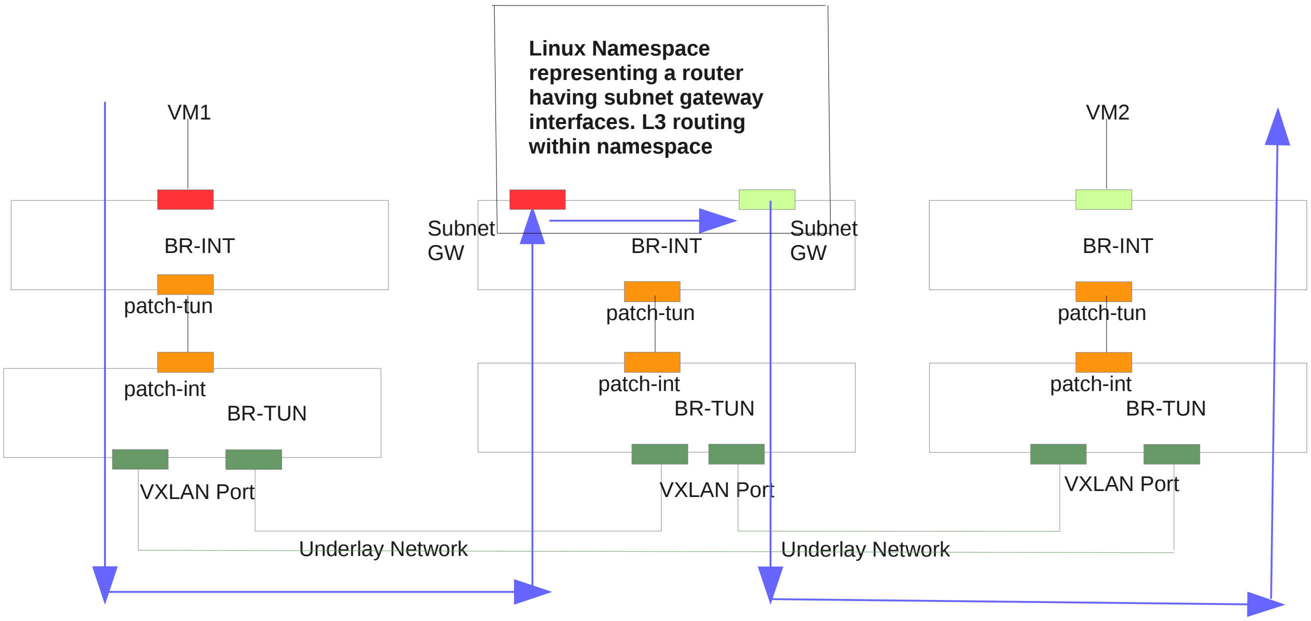


FIG 5.1 - ORIGINAL MECHANISM

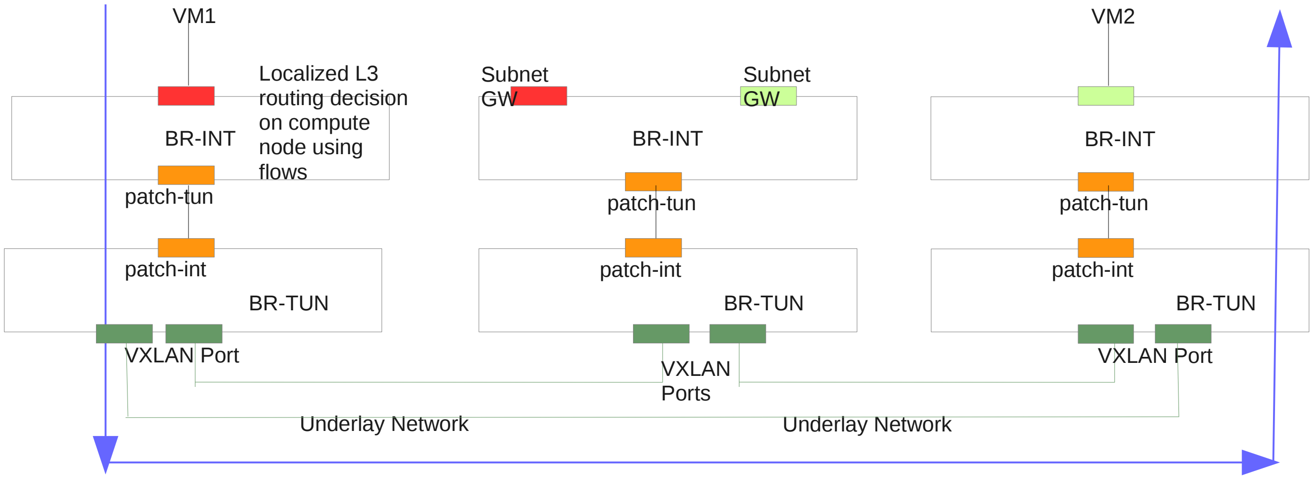


Fig 5.2- NEW MECHANISM

- L3 Routing / Network Address Translations (NAT) using flows on OVS external bridge

SNAT / DNAT completely using flows defined on OVS external bridge against the existing mechanism of using Linux Namespaces, iptables rule, Host TCP/IP Stack and conntrack utility. Thus, there is no dependency on Host TCP/IP stack for NATing and maintaining sessions for external network. For the POC, we use port-forwarding for SNAT/DNAT purpose. It thus maintains, L4 sessions on OVS external for TCP/UDP packets. ICMP packets can be also SNATed using flows defined on external bridge, however, flows currently provide access to only ICMP type and ICMP code which are not enough to uniquely identify each session. Thus, unambiguous support for SNAT of ICMP packets can be supported in future. (Refer Fig. 6)

However, there are certain restrictions imposed due to non-availability of access to ICMP identifier field by the OpenFlow Protocol mentioned in “FUNCTIONALITIES SUPPORTED” section

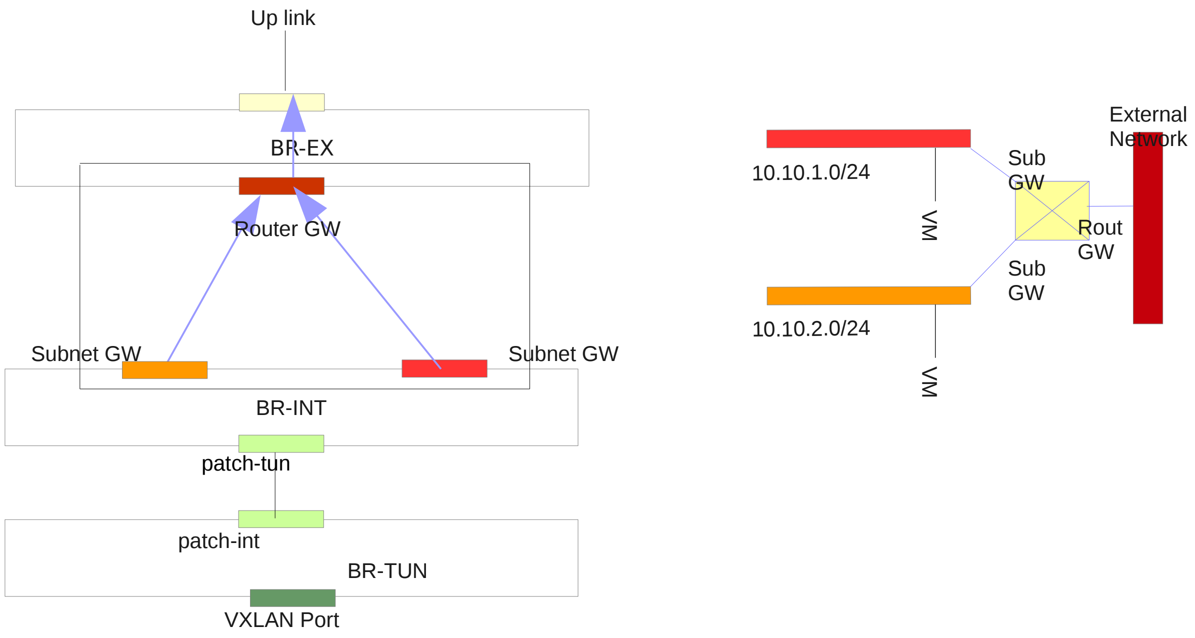


Fig 6.1- NETWORK NODE (OLD MECHANISM)

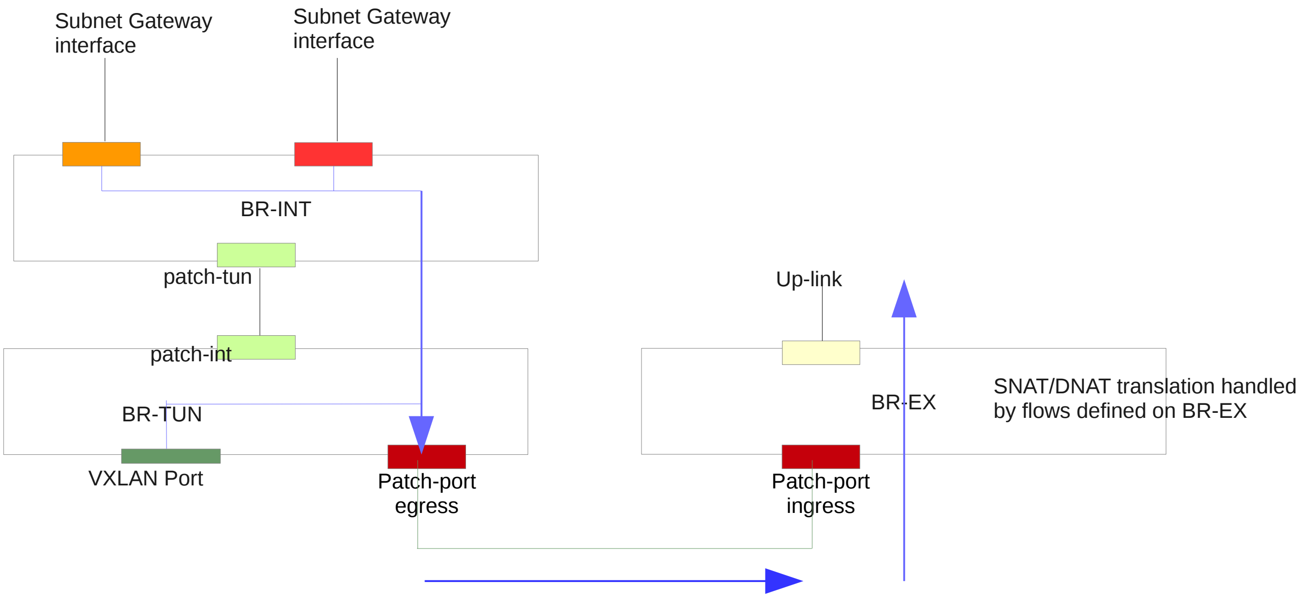


Fig 6.2- NETWORK NODE (NEW MECHANISM)

- Single instance of the new agent for multiple external networks

A single instance of the new agent developed as a part of POC handles SNAT/DNAT for multiple external networks against the existing mechanism (OpenStack HAVANA) of using one neutron-l3-agent per external network on the network node. The same agent can be scaled to use multiple external networks on same OVS external bridge or one external network per OVS external bridge. The existing setup uses one OVS external bridge per external network. (Refer Fig. 7)

The POC currently can uses one OVS external bridge only and can have multiple uplinks mapped to it. POC can be scaled / modified to distribute up-links on multiple external bridges and handle them separately using single instance of the common agent.

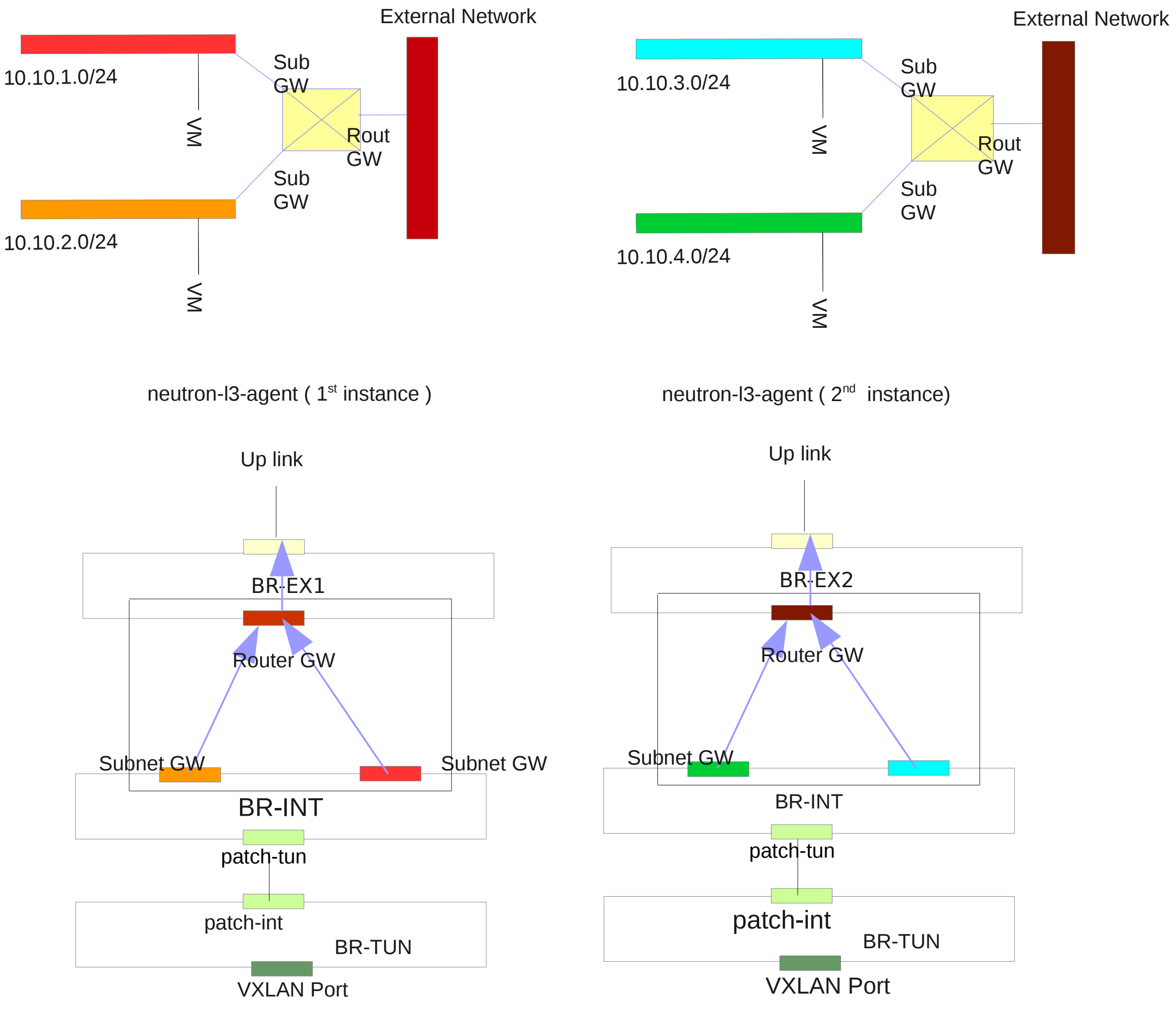
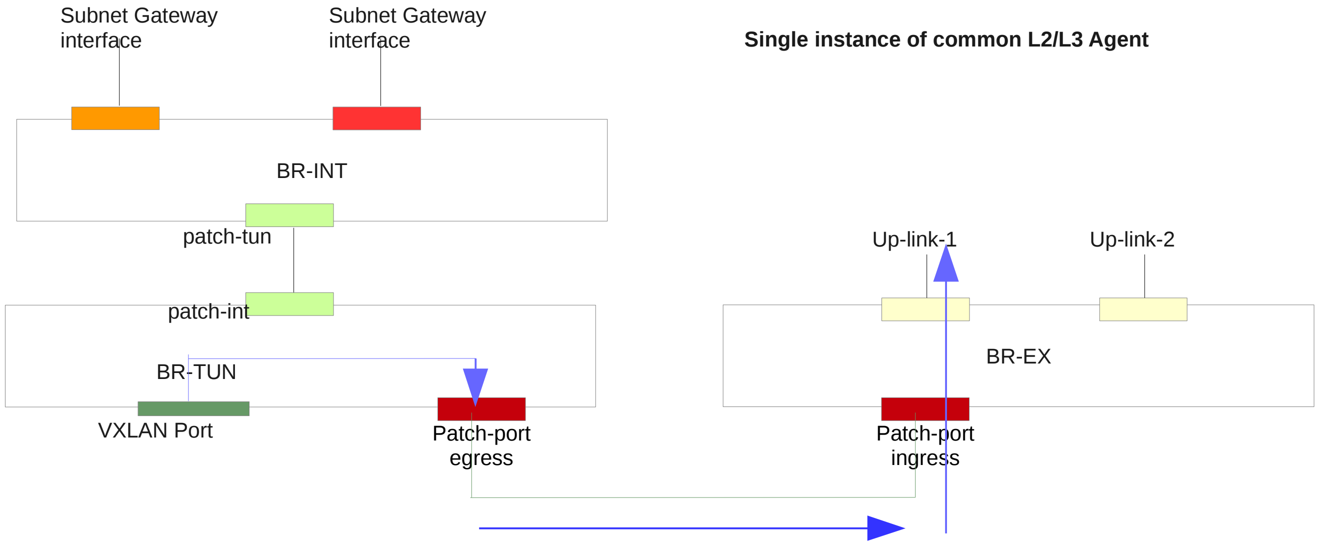


Fig 7.1- NETWORK NODE (OLD MECHANISM)



Single instance of common L2/L3 Agent

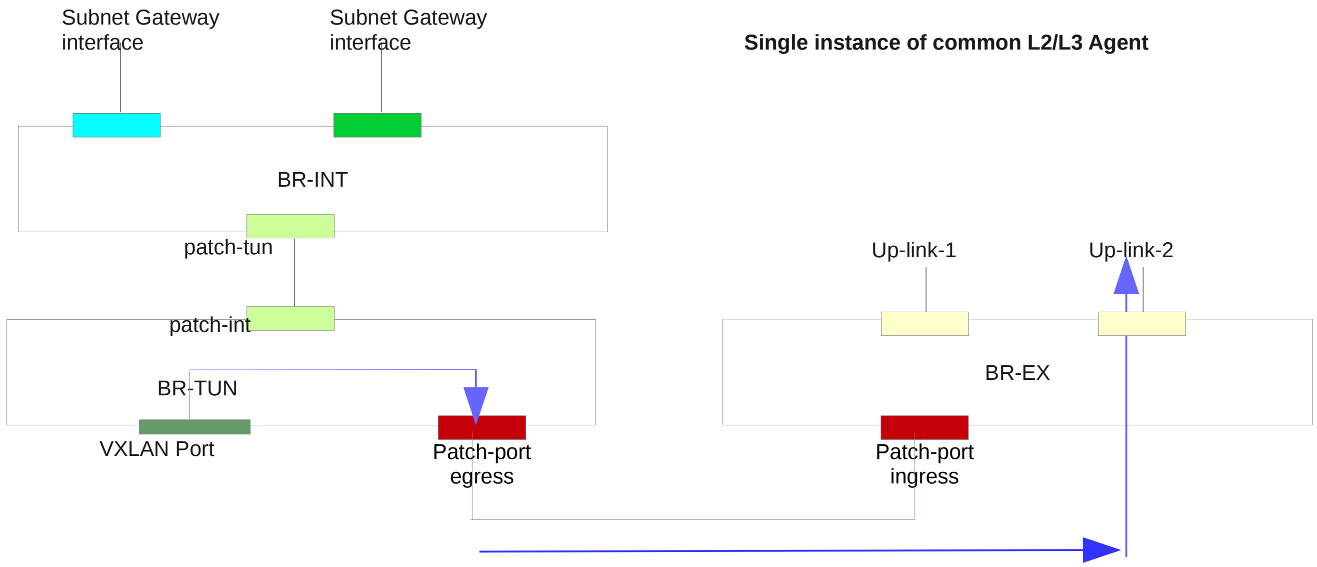
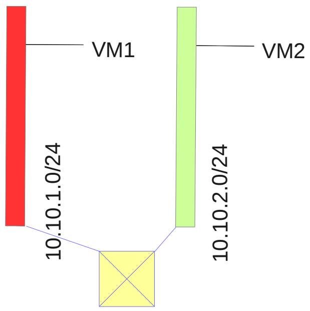


Fig 7.2- NETWORK NODE (NEW MECHANISM)

- L2 Population and ARP Responder Mechanism
POC makes use of L2 population and ARP responder mechanism introduced in OpenStack Havana. L2 population reduces unnecessary broadcasts in the underlay network by reducing the L2 broadcasting domains amongst the VXLAN Tunnel Ports.

ARP Responder mechanism is used on each compute node to reduce the flooding of ARP Packets in the underlay network. Thus, ARP replies are generated on each host via flows defined on the OVS Tunnel. (Refer Fig. 8)

ARP Responder mechanism is also introduced on the OVS external bridge for handling ARP Request from underlay network for SNAT / DNAT ports on Neutron Router. Thus OVS external bridge generates ARP Replies for the underlay network completely via flows.



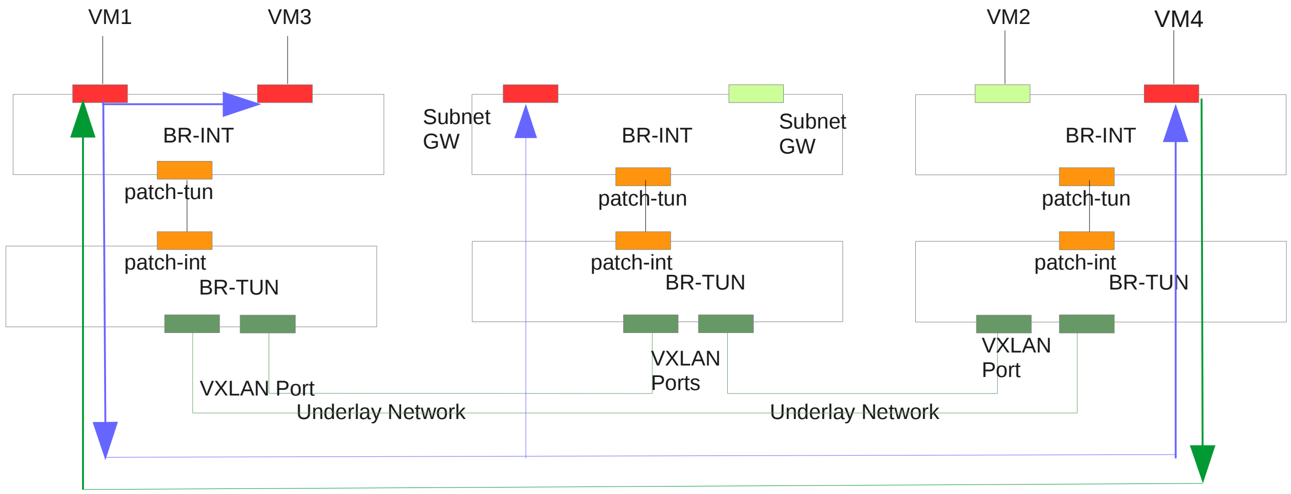


Fig 8.1- OLD MECHANISM

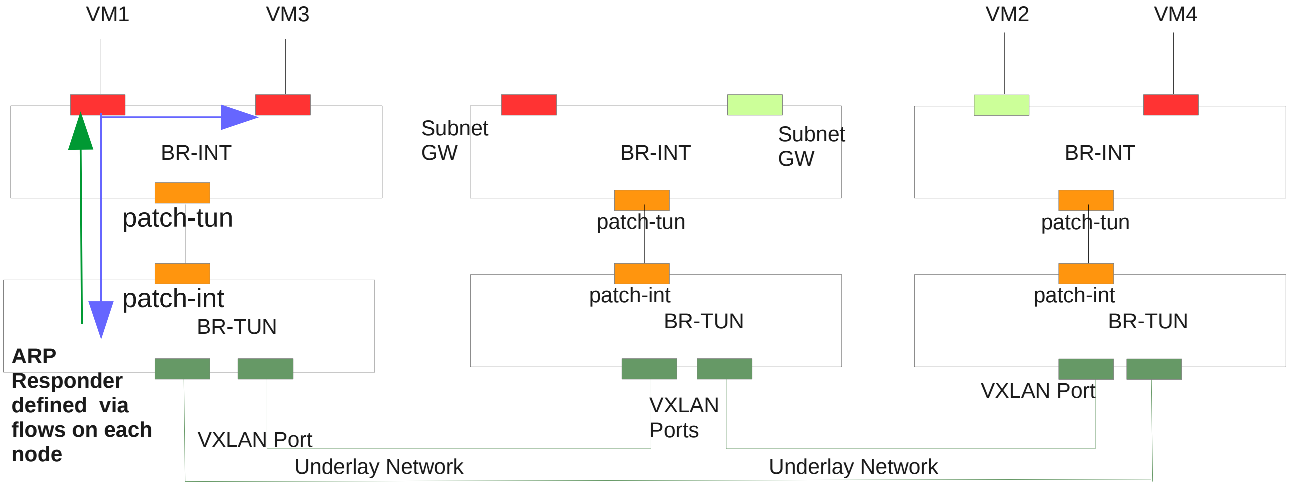


Fig 8.2- NEW MECHANISM

- Routing amongst VM's on networks connected by different virtual routers
The POC supports routing amongst different routers in the overlay, with certain limitations constrained by OpenFlow protocol for ICMP sessions mentioned in "FUNCTIONALITIES SUPPORTED" section

- L2 unicast / broadcast using flows defined on OVS Integration and Tunnel Bridge
Flows are defined on OVS integration Bridge for completely handling L2 unicast / broadcast and VLAN Tagging / Untagging against the existing mechanism of using OVS integration as a normal L2 Switch (NORMAL action) and relying on OVSDB. Transmission of packets across host is handled by flows defined on OVS Tunnel.

- Port Optimization / MAC Optimization
The POC does port optimization with respect to router ports (subnet gateway, router gateway ports). Since all the routing decisions are now handled via flows on OVS bridges, the POC does not actually need to create any port on OVS integration for subnet gateways and on OVS external for router gateways. Thus, achieving port optimization in the process.

The Distributed Virtual Router (DVR) functionality introduced in OpenStack Juno requires one MAC address per compute node. The POC requires no such additional MAC for the purpose. Also, the Opensatck Setup currently uses one MAC per subnet gateway interface. The POC can be easily modified to use one global MAC throughout for multiple subnet gateways and still unambiguously handle overlay across subnet and overlay to external routing. Thus, providing MAC optimization in the process

- Support for Multi-Tenant Environment
The POC is fully compatible with the multi-tenant environment

- MAC repetition across networks
The POC supports MAC Address repetition across networks but not within a network

FUTURE WORK

- SNAT support for ICMP packets (Ping)
Flows defined by OpenFlow Protocol currently support access to only ICMP type and code and not ICMP Identifier field, and thus, cannot be used for unambiguous SNATing (SNAT requires identifier field). Thus introducing access to ICMP identifier by the flows can help in achieving SNAT and maintain each unique SNAT session and overcome the existing shortcomings relating to ICMP SNATing.

- Port Mapping instead of Port Forwarding for ICMP / IP sessions from overlay to underlay
The POC currently does a direct port forwarding leading to chances of port collisions. This shortcoming can be solved by sending the first packet to the agent and doing Port Mapping and adding the flows on OVS external bridge accordingly for subsequent packets thus avoiding any port collision.

- Support for ICMP Error Messages
The POC currently does not provision for any ICMP Error Messages.

- ICMP packets (Ping) support for subnet gateway and router gateway (SNAT) interfaces on router
The POC currently does not support ICMP Packets (Ping) for router interfaces. These can be supported by forwarding the ICMP requests to the new agent and provisioning the agent to generate the ICMP replies. Alternative way could be to develop ICMP Echo reply mechanism in flows on Open vSwitch just like ARP Responder mechanism

- Support multiple external networks / uplinks on single OVS external bridge programatically
Currently the user needs to add flows manually to support multiple up-links on a single OVS external bridge. The agent in POC currently programatically adds flows for one uplink on the bridge. This process can be automated by the agent for multiple uplinks

- Implement OpenStack Nova Security Groups using flows in OVS integration bridge
OpenStack Nova Security groups are currently implemented using iptables which are not compatible with OVS bridge. Because of this the VM’s are not directly mapped on to the OVS Integration Bridge and have an intermediate Linux Bridge. Security Groups can be implemented using flows on OVS integration Bridge allowing the VM’s to be directly mapped on to the OVS Integration Bridge. (refer Fig 8)

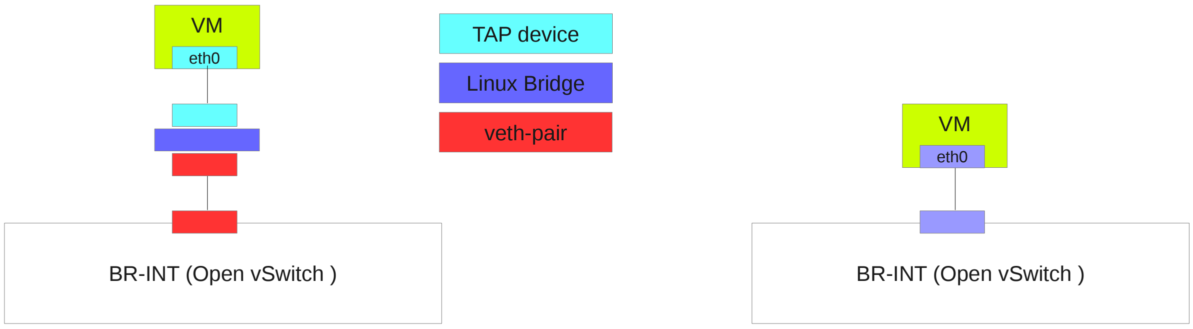


Fig 8.1- CURRENT SETUP

Fig 8.2- NEW SETUP

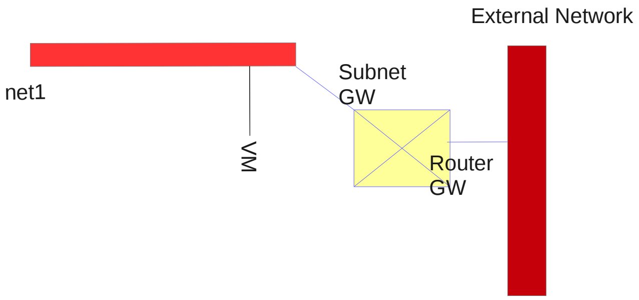
- Updation of ARP Cache for underlay network nodes on OVS External Bridge programmatically
Flows relating to ARP entries for nodes in underlay network needs to be added manually cuurently in the POC. Thus, mechanism can be developed to populate the ARP cache dynamically by the agent

- Updation of ARP Cache for overlay network ports on OVS Integration / Tunnel Bridge programmatically
ARP Cache entries for ports are currently static in nature in POC i.e. they do not timeout. Thus, mechanism can be developed to populate the ARP cache dynamically by the agent.

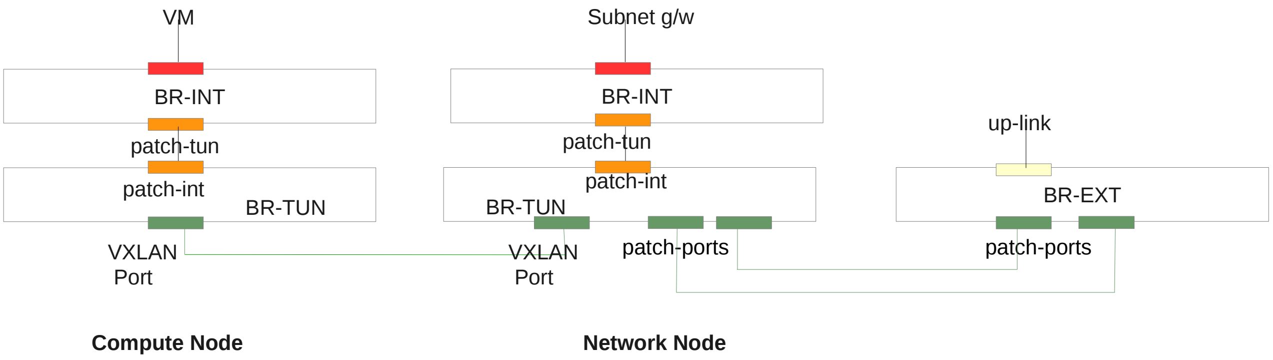
- Deletion of flows in a programmatic manner
The POC deals with only creation and addition of flows on the OVS bridges. POC can be extended to support deletion of flows for a stand alone agent.

USE CASE

This sections presents a a sample use case for SNAT for the following topology



A 2 node setup, having 1 compute and 1 network-controller node.



1 network having 1 subnet
Overlay IP for VM is VM_IP and MAC address of VM is VM_MAC
Subnet gateway IP is SUBNET_GW_IP with MAC address of SUBNET_GW_MAC
Router gteway IP is ROUTER_GW_IP with MAC address ROUTER_GW_MAC
Overlay internal network VLAN tag- VLAN_INTERNAL
Underlay external network VLAN tag- VLAN_EXTERNAL
Underlay external network VXLAN tag- VXLAN_EXTERNAL

Lets say that the VM ping an external node having IP- 9.121.62.66

Following are the major sub-steps-

- ARP request for subnet gateway by VM
- ARP Reply for subnet gateway to VM by ARP Responder Mechanism
- ICMP request for 9.121.62.66 from VM on compute node sent to network node
- SNATing at Network Node for ICMP Request
- NATing to overlay IP and MAC on OVS External Bridge for ICMP reply from 9.121.62.66 received on OVS external bridge
- Forwarding to the VM on compute node

We will now discuss each step in details, shows the OpenFlow pipeline followed on each bridge and the MAC / IP / VLAN / VXLAN translations

Step 1 - ARP request for subnet gateway by VM
The process of sending any ICMP packet to 9.121.62.66 is first preceded by generating an ARP Request for subnet gateway by the VM. The VM thus generates an ARP reply for the subnet gateway which is transferred from VM patch to the OVS Tunnel bridge via patch-port after VLAN Tagging in table2

Source MAC- VM_MAC	**Dest MAC**- ff:ff:ff:ff:ff:ff	**ARP_SPA**- VM_IP,
ARP_TP A- SUBNET_GW_IP	**ARP_SHA**- VM_MAC	**ARP_THA**- 00:00:00:00:00:00

VLAN_TAG- INTERNAL_VLAN

In the process, a learned flow is created in Table 20 so that the ARP reply can be sent to the VM directly

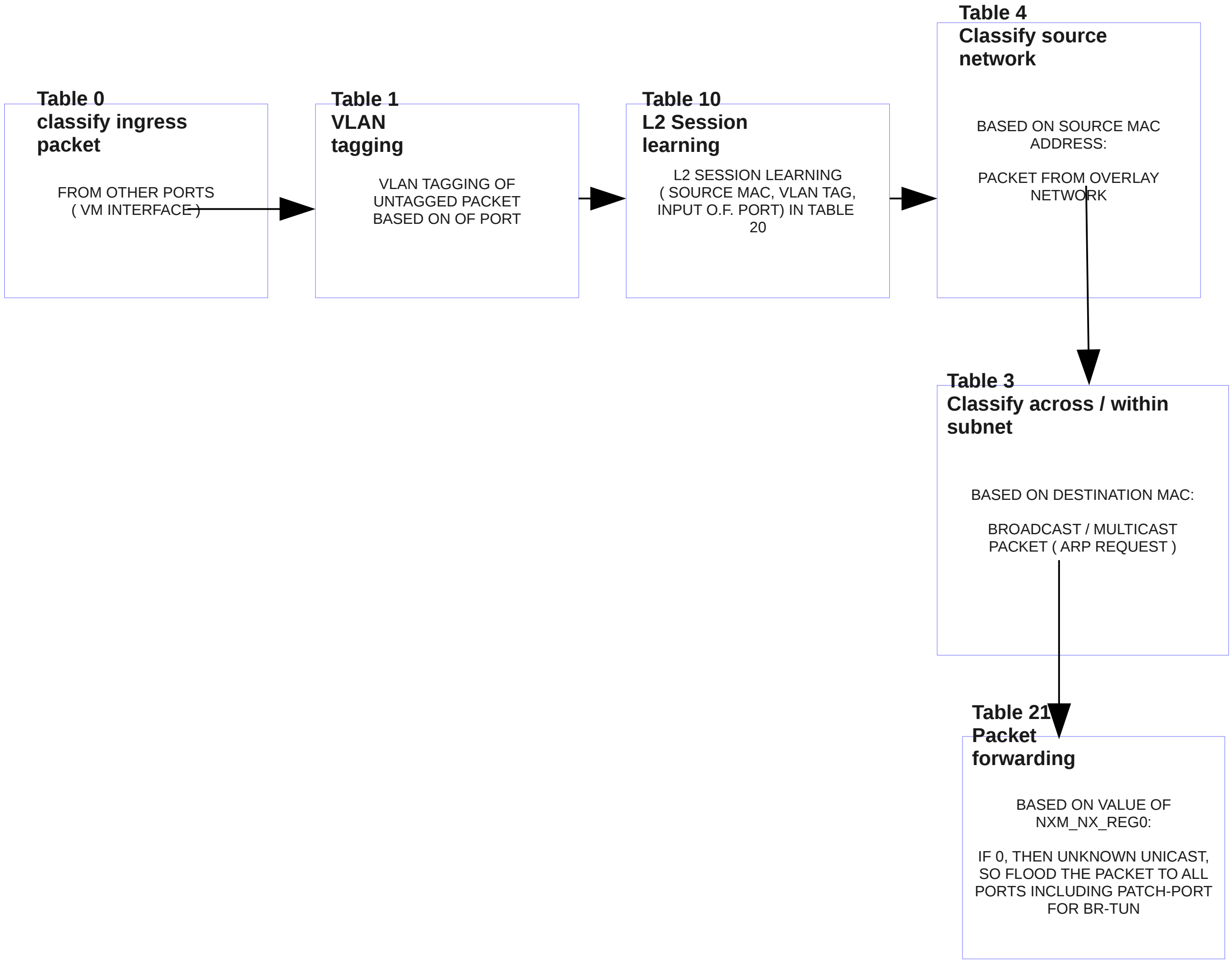
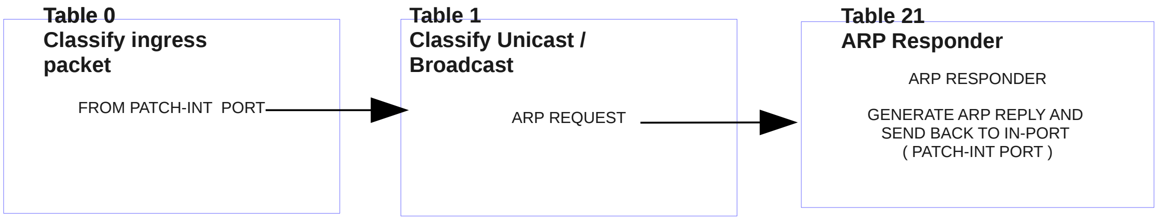


Table 0
classify ingress packet
FROM OTHER PORTS (VM INTERFACE)
Table 1
VLAN tagging
VLAN TAGGING OF UNTAGGED PACKET BASED ON OF PORT
Table 10
L2 Session learning
L2 SESSION LEARNING (SOURCE MAC, VLAN TAG, INPUT O.F. PORT) IN TABLE 20
Table 4
Classify source network
BASED ON SOURCE MAC ADDRESS:
PACKET FROM OVERLAY NETWORK
Table 3
Classify across / within subnet
BASED ON DESTINATION MAC:
BROADCAST / MULTICAST PACKET (ARP REQUEST)
Table 21
Packet forwarding
BASED ON VALUE OF NXM_NX_REG0:
IF 0, THEN UNKNOWN UNICAST, SO FLOOD THE PACKET TO ALL PORTS INCLUDING PATCH-PORT FOR BR-TUN

STEP 2- ARP Reply for subnet gateway to VM by ARP Responder Mechanism
On receiving an ARP request from the OVS integration patch-port, OVS Tunnel Bridge, generates and ARP Reply for the subnet gateway in table 21

Source MAC- SUBNET_GW_MAC **Dest MAC-** VM_MAC **ARP_SPA-** SUBNET_GW_IP
ARP_TP A- VM_IP **ARP_SHA-** SUBNET_GW_MAC **ARP_THA-** VM_MAC
VLAN_TAG- INTERNAL_VLAN

and sends it back to the input port (OVS integration patch-port). On receiving the ARP reply (unicast) on the patch-port in OVS integration bridge, the flows refer to the learned flow in Table 20 and forward the packet to the VM



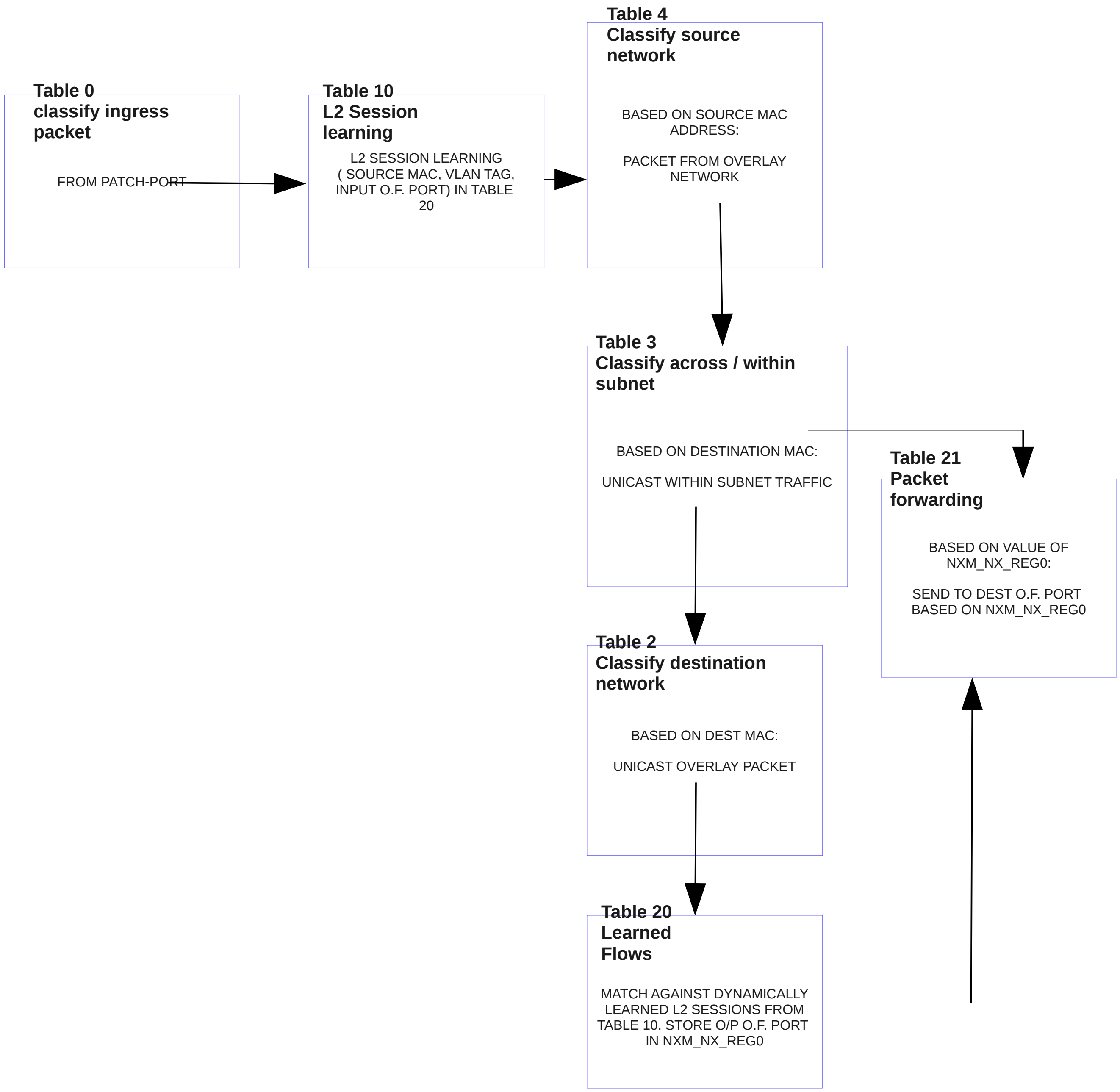


Table 0
classify ingress packet
FROM PATCH-PORT
Table 10
L2 Session learning
L2 SESSION LEARNING (SOURCE MAC, VLAN TAG, INPUT O.F. PORT) IN TABLE 20
Table 4
Classify source network
BASED ON SOURCE MAC ADDRESS:
PACKET FROM OVERLAY NETWORK
Table 3
Classify across / within subnet
BASED ON DESTINATION MAC:
UNICAST WITHIN SUBNET TRAFFIC
Table 21
Packet forwarding
BASED ON VALUE OF NXM_NX_REG0:
SEND TO DEST O.F. PORT BASED ON NXM_NX_REG0
Table 2
Classify destination network
BASED ON DEST MAC:
UNICAST OVERLAY PACKET
Table 20
Learned Flows
MATCH AGAINST DYNAMICALLY LEARNED L2 SESSIONS FROM TABLE 10. STORE O/P O.F. PORT IN NXM_NX_REG0

STEP 3- ICMP request for 9.121.62.66 from VM on compute node sent to network node
After receiving the ARP reply for subnet gateway, the VM sends an ICMP echo request for 9.121.62.66.

Source MAC- VM_MAC **Dest MAC**- SUBNET_GW_MAC **Source IP**- VM_IP,
Dest IP - 9.121.62.66

The flows on OVS integration bridge handle the VLAN tagging of packet for the source network of which VM is a part of in table=1

Source MAC- VM_MAC **Dest MAC**- SUBNET_GW_MAC **Source IP**- VM_IP,
Dest IP - 9.121.62.66 **VLAN** - INTERNAL_VLAN

A learning flow is created in Table 20 for the ICMP reply from 9.121.62.66. The flows then transform the destination MAC from subnet gateway MAC to router gateway MAC and change the VLAN tag to that of the external network in Table 50 and forward the packet to patch-port

Source MAC- SUBNET_GW_MAC **Dest MAC**- ROUTER_GW_MAC **Source IP**- VM_IP
Dest IP- 9.121.62.66 **VLAN**- EXTERNAL_VLAN

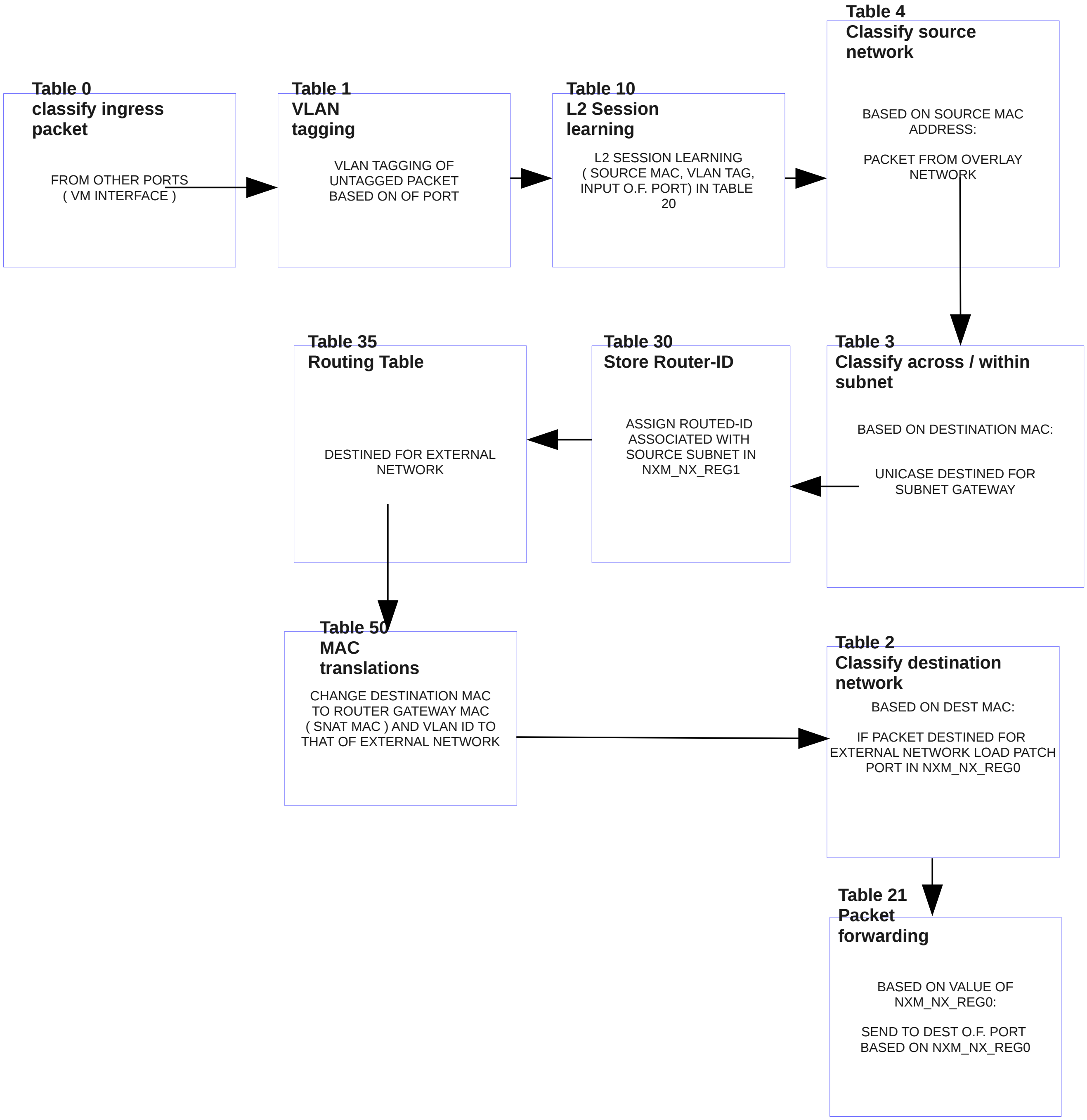


Table 0
classify ingress packet
FROM OTHER PORTS (VM INTERFACE)
Table 1
VLAN tagging
VLAN TAGGING OF UNTAGGED PACKET BASED ON OF PORT
Table 10
L2 Session learning
L2 SESSION LEARNING (SOURCE MAC, VLAN TAG, INPUT O.F. PORT) IN TABLE 20
Table 4
Classify source network
BASED ON SOURCE MAC ADDRESS:
PACKET FROM OVERLAY NETWORK
Table 3
Classify across / within subnet
BASED ON DESTINATION MAC:
UNICASE DESTINED FOR SUBNET GATEWAY
Table 30
Store Router-ID
ASSIGN ROUTED-ID ASSOCIATED WITH SOURCE SUBNET IN NXM_NX_REG1
Table 35
Routing Table
DESTINED FOR EXTERNAL NETWORK
Table 50
MAC translations
CHANGE DESTINATION MAC TO ROUTER GATEWAY MAC (SNAT MAC) AND VLAN ID TO THAT OF EXTERNAL NETWORK
Table 2
Classify destination network
BASED ON DEST MAC:
IF PACKET DESTINED FOR EXTERNAL NETWORK LOAD PATCH PORT IN NXM_NX_REG0
Table 21
Packet forwarding
BASED ON VALUE OF NXM_NX_REG0:
SEND TO DEST O.F. PORT BASED ON NXM_NX_REG0

On receiving the packet from patch port in OVS tunnel bridge, the packet is forwarded to the network node directly by removing VLAN tag and loading the VXLAN ID for external network in Table 31

Source MAC- SUBNET_GW_MAC **Dest MAC**- ROUTER_GW_MAC **Source IP**- VM_IP
Dest IP- 9.121.62.66 **VxLAN**- EXTERNAL_VXLAN

The packet is then forwarded to the VXLAN tunnel port corresponding to the network node

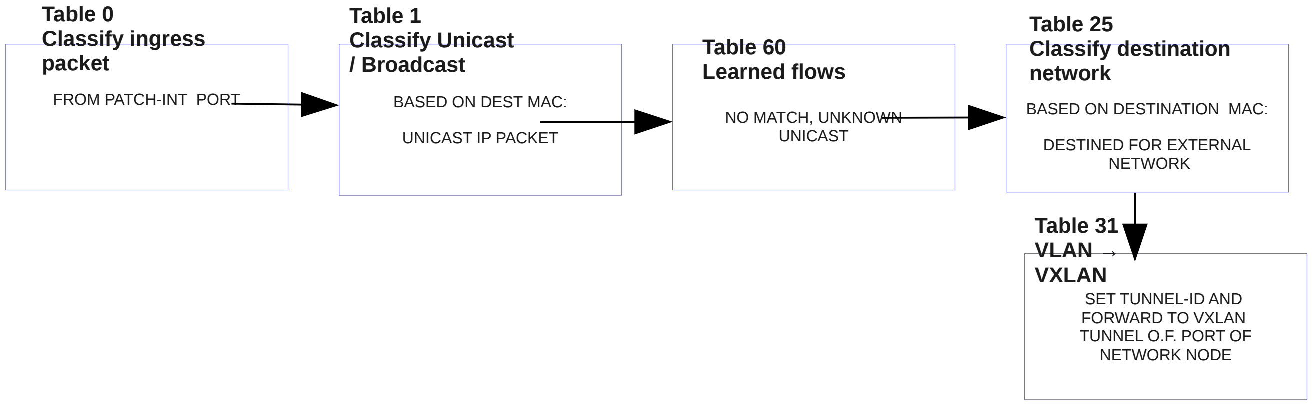


Table 0
Classify ingress packet
FROM PATCH-INT PORT
Table 1
Classify Unicast / Broadcast
BASED ON DEST MAC:
UNICAST IP PACKET
Table 60
Learned flows
NO MATCH, UNKNOWN UNICAST
Table 25
Classify destination network
BASED ON DESTINATION MAC:
DESTINED FOR EXTERNAL NETWORK
Table 31
VLAN → VXLAN
SET TUNNEL-ID AND FORWARD TO VXLAN TUNNEL O.F. PORT OF NETWORK NODE

STEP 4- SNATing at Network Node for ICMP Request
The packet is received by OVS tunnel bridge on the Network Node which handles VLAN tagging of the packet to that of the external network using VXLAN ID in table 3. A flow is created in Table 61 for the L3 session which handles ICM reply from 9.121.62.66 and the packet is sent to the OVS external bridge

(Packet after encountering table 3)

Source MAC- SUBNET_GW_MAC	**Dest MAC**- ROUTER_GW_MAC	**Source IP**- VM_IP
Dest IP- 9.121.62.66	**VLAN**- EXTERNAL_VLAN	

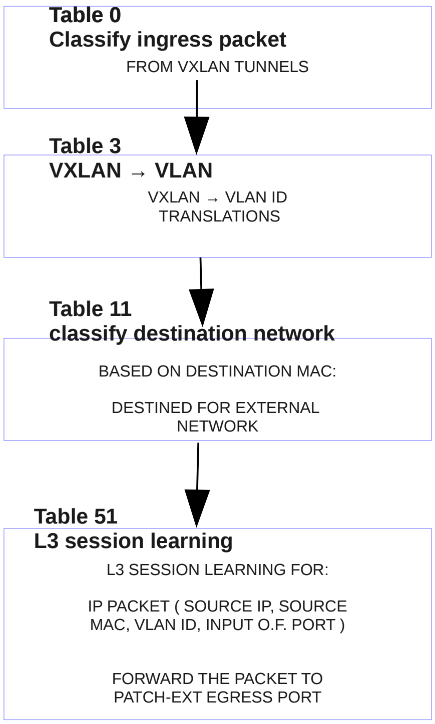


Table 0
Classify ingress packet
FROM VXLAN TUNNELS
Table 3
VXLAN → VLAN
VXLAN → VLAN ID TRANSLATIONS
Table 11
classify destination network
BASED ON DESTINATION MAC:
DESTINED FOR EXTERNAL NETWORK
Table 51
L3 session learning
L3 SESSION LEARNING FOR:
IP PACKET (SOURCE IP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
FORWARD THE PACKET TO PATCH-EXT EGRESS PORT

The packet is received on the OVS external bridge. Since there no floatingIP associated with VM, pipeline for SNATIng is followed. First a flow is created table 40 for ICMP reply then VLAN Tag is stripped from the packet and NATing is done for the outgoing ICMP request in table 30.

(Packet after encountering table 30)

Source MAC- ROUTER_GW_MAC	**Dest MAC**- ROUTER_GW_MAC	**Source IP**- SNAT_IP
Dest IP- 9.121.62.66		

The packet is then forwarded 9.121.62.66 by changing the destination MAC to that of 9.121.62.66 in table 20

Source MAC- ROUTER_GW_MAC	**Dest MAC**- MAC_OF_9.121.62.66	**Source IP**- SNAT_IP

Dest IP- 9.121.62.66
The packet is now forwarded to the up-link port

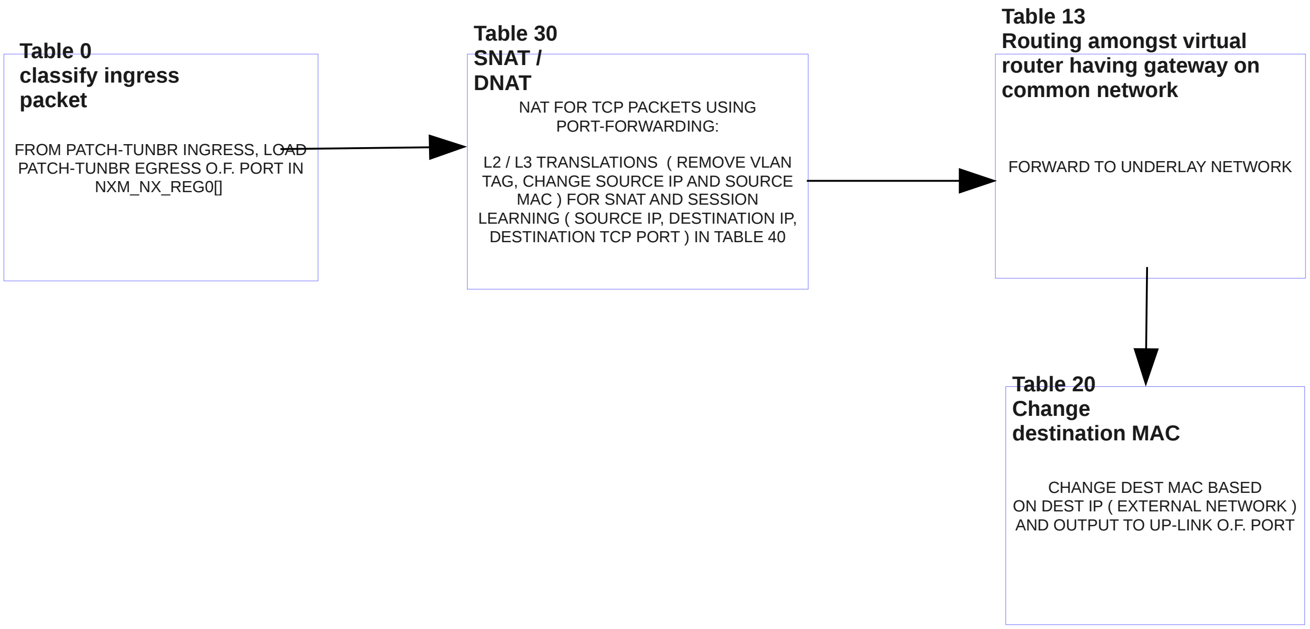


Table 0
classify ingress
packet
FROM PATCH-TUNBR INGRESS, LOAD PATCH-TUNBR EGRESS O.F. PORT IN NXM_NX_REG0[]
Table 30
SNAT /
DNAT
NAT FOR TCP PACKETS USING PORT-FORWARDING:
L2 / L3 TRANSLATIONS (REMOVE VLAN TAG, CHANGE SOURCE IP AND SOURCE MAC) FOR SNAT AND SESSION LEARNING (SOURCE IP, DESTINATION IP, DESTINATION TCP PORT) IN TABLE 40
Table 13
Routing amongst virtual router having gateway on common network
FORWARD TO UNDERLAY NETWORK
Table 20
Change
destination MAC
CHANGE DEST MAC BASED ON DEST IP (EXTERNAL NETWORK) AND OUTPUT TO UP-LINK O.F. PORT

STEP 5- NATing to overlay IP and MAC on OVS External Bridge for ICMP reply from 9.121.62.66 received on OVS external bridge

Following is the IP/MAC address of the ICMP reply received on the up-link on OVS external bridge

Source MAC- MAC_OF_9.121.62.66 **Dest MAC**- ROUTER_GW_MAC **Source IP**- 9.121.62.66
Dest IP- SNAT_IP

On receiving the ICMP reply from 9.121.62.66, the learned flows in Table 40 handle the IP/MAC translation from underlay MAC/IP to overlay MAC/IP and VLAN Tagging to that of the external network.

Source MAC- ROUTER_GW_MAC **Dest MAC**- SUBNET_GW_MAC **Source IP**- 9.121.62.66
Dest IP- VM_IP **VLAN-** EXTERNAL_VLAN

The packet is then sent to OVS Tunnel bridge

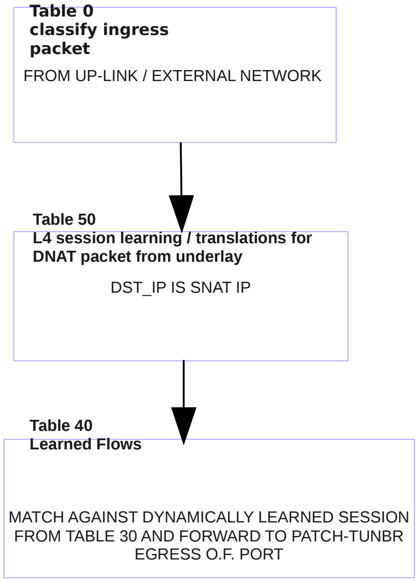


Table 0
classify ingress
packet
FROM UP-LINK / EXTERNAL NETWORK
Table 50
L4 session learning / translations for DNAT packet from underlay
DST_IP IS SNAT IP
Table 40
Learned Flows
MATCH AGAINST DYNAMICALLY LEARNED SESSION FROM TABLE 30 AND FORWARD TO PATCH-TUNBR EGRESS O.F. PORT

The packet on being received by OVS tunnel from OVS external bridge encounters learned flow in Table 61 for ICMP request and is sent directly to the compute node hosting the destination VM after removing VLAN Tag and allocating VXLAN ID for the external network

(Packet after encountering table 61)
Source MAC- ROUTER_GW_MAC **Dest MAC**- SUBNET_GW_MAC **Source IP-** 9.121.62.66
Dest IP- VM_IP **VxLAN-** EXTERNAL_VXLAN

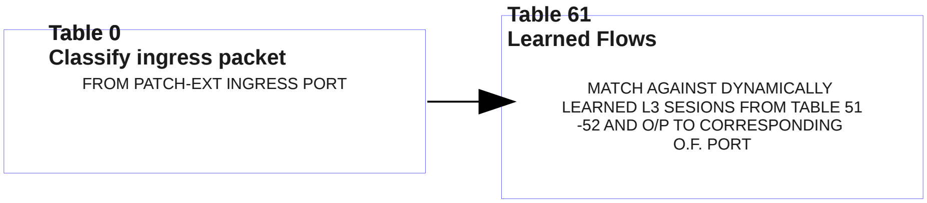


Table 0
Classify ingress packet
FROM PATCH-EXT INGRESS PORT
Table 61
Learned Flows
MATCH AGAINST DYNAMICALLY LEARNED L3 SESIONS FROM TABLE 51 -52 AND O/P TO CORRESPONDING O.F. PORT

STEP 6 - Forwarding to the VM on compute node
The packet is received by OVS Tunnel on the compute nodes. The packet is VLAN tagged to that of the external network in table 3

Source MAC- ROUTER_GW_MAC **Dest MAC**- SUBNET_GW_MAC **Source IP-** 9.121.62.66
Dest IP- VM_IP **VLAN-** EXTERNAL_VLAN

The packet is than forwarded to the patch port corresponding to the OVS Integration Bridge.

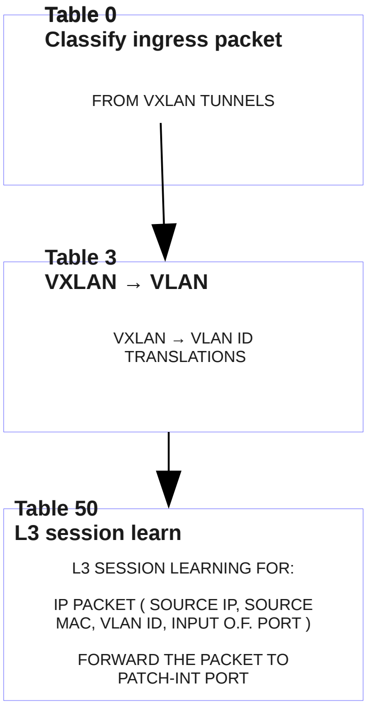


Table 0
Classify ingress packet
FROM VXLAN TUNNELS
Table 3
VXLAN → VLAN
VXLAN → VLAN ID TRANSLATIONS
Table 50
L3 session learn
L3 SESSION LEARNING FOR:
IP PACKET (SOURCE IP, SOURCE MAC, VLAN ID, INPUT O.F. PORT)
FORWARD THE PACKET TO PATCH-INT PORT

On receiving the packet on OVS integration bridge, the packets source MAC address is changed from router gateway MAC to subnet gateway MAC in table 5, along with removing VLAN tag for external network and adding VLAN tag for the network of destination VM.

Source MAC- SUBNET_GW_MAC **Dest MAC**- VM_MAC **Source IP-** 9.121.62.66
Dest IP- VM_IP **VLAN-** INTERNAL_VLAN

The packet is then referred to table 20 which contains the flows for forwarding the packet directly to VM.

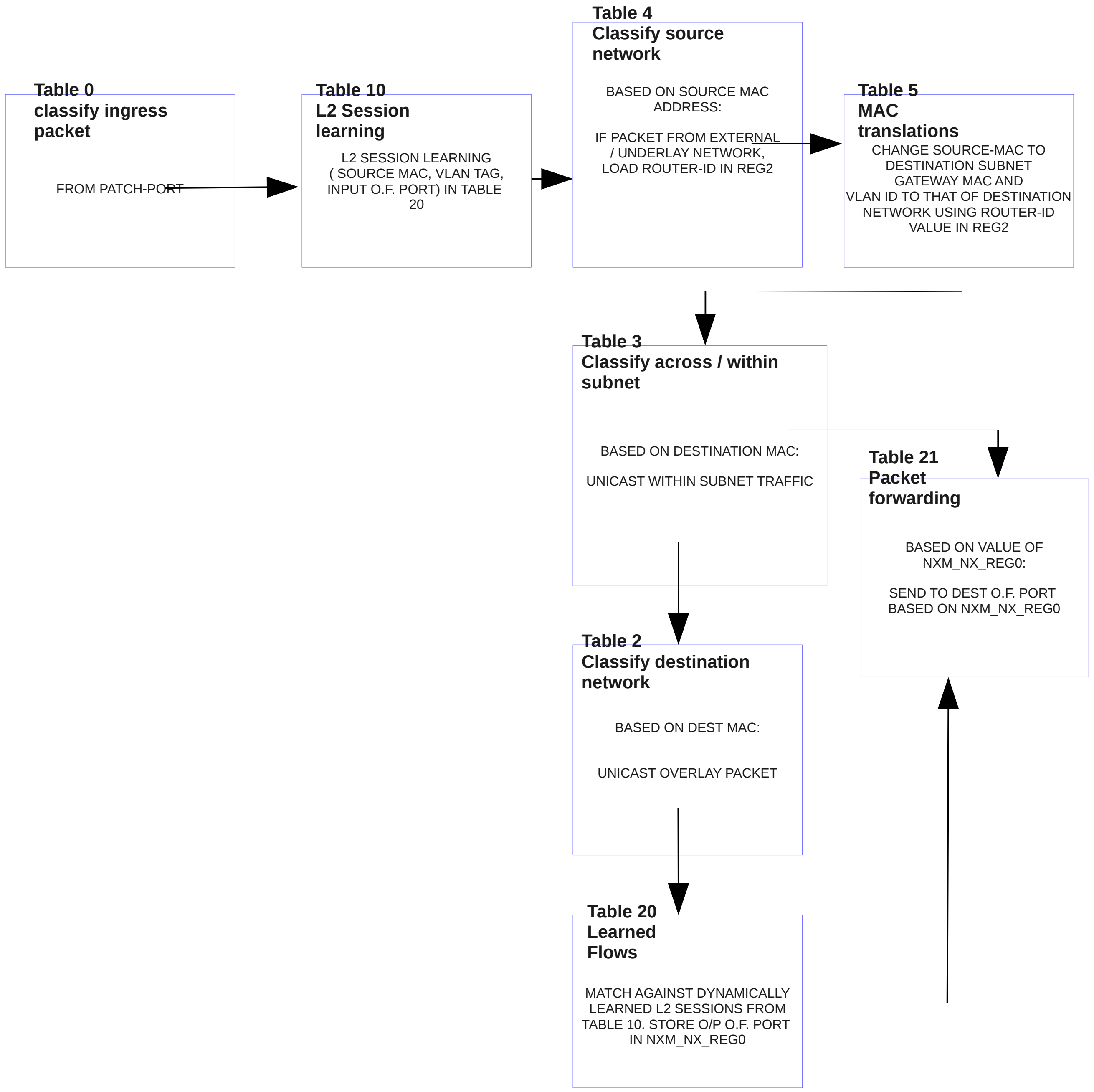
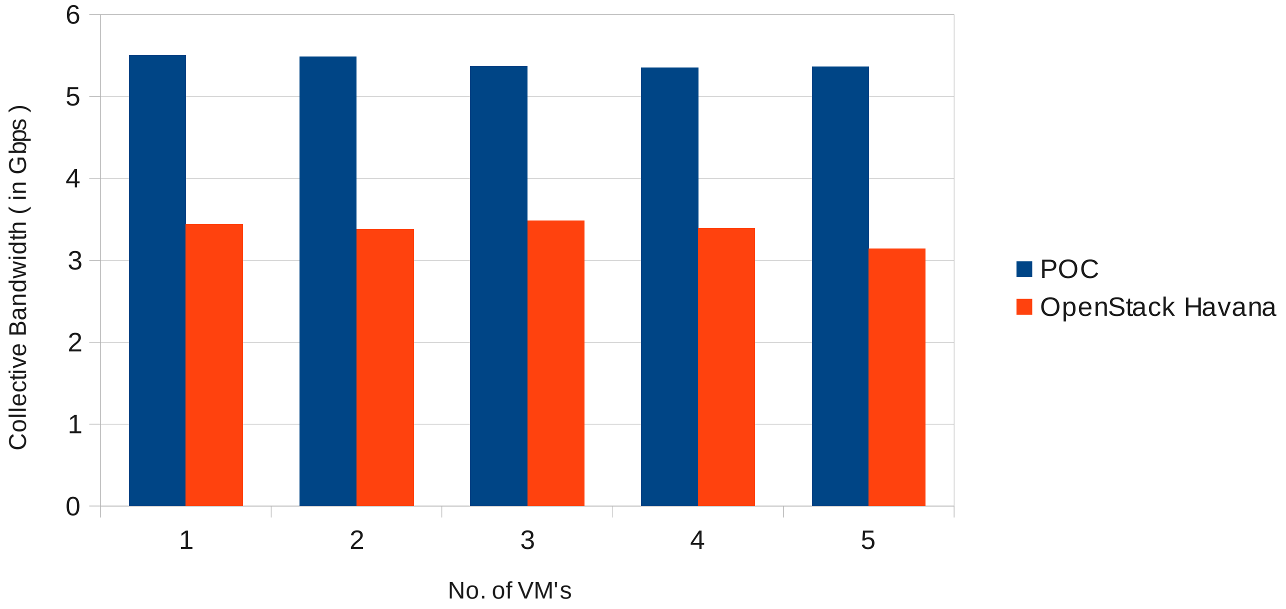


Table 0
classify ingress packet
FROM PATCH-PORT
Table 10
L2 Session learning
L2 SESSION LEARNING (SOURCE MAC, VLAN TAG, INPUT O.F. PORT) IN TABLE 20
Table 4
Classify source network
BASED ON SOURCE MAC ADDRESS:
IF PACKET FROM EXTERNAL / UNDERLAY NETWORK, LOAD ROUTER-ID IN REG2
Table 5
MAC translations
CHANGE SOURCE-MAC TO DESTINATION SUBNET GATEWAY MAC AND VLAN ID TO THAT OF DESTINATION NETWORK USING ROUTER-ID VALUE IN REG2
Table 3
Classify across / within subnet
BASED ON DESTINATION MAC:
UNICAST WITHIN SUBNET TRAFFIC
Table 21
Packet forwarding
BASED ON VALUE OF NXM_NX_REG0:
SEND TO DEST O.F. PORT BASED ON NXM_NX_REG0
Table 2
Classify destination network
BASED ON DEST MAC:
UNICAST OVERLAY PACKET
Table 20
Learned Flows
MATCH AGAINST DYNAMICALLY LEARNED L2 SESSIONS FROM TABLE 10. STORE O/P O.F. PORT IN NXM_NX_REG0

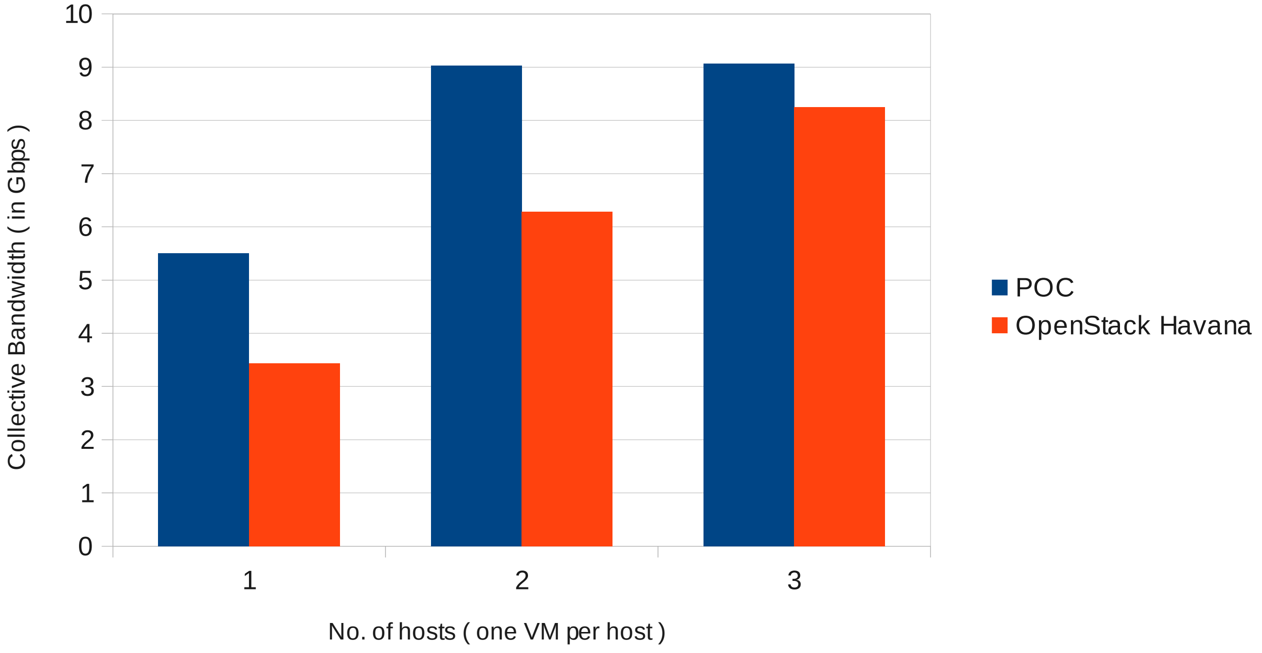
PERFORMANCE ANALYSIS

This section presents the performance analysis of the POC against the existing Openstack Havana Setup for various use cases. For the performance analysis we used a 4 node setup (1 Network-Controller and 3 Compute Nodes). All the up-links were 10 Gbps uplinks (VxLAN tunnel port on each node, external network up-link on Network Node and the up-link on the node in the external network) Following was the VM configuration used- 2 VCPUs, 2048 MB RAM, 20 GB HDD, CentOS

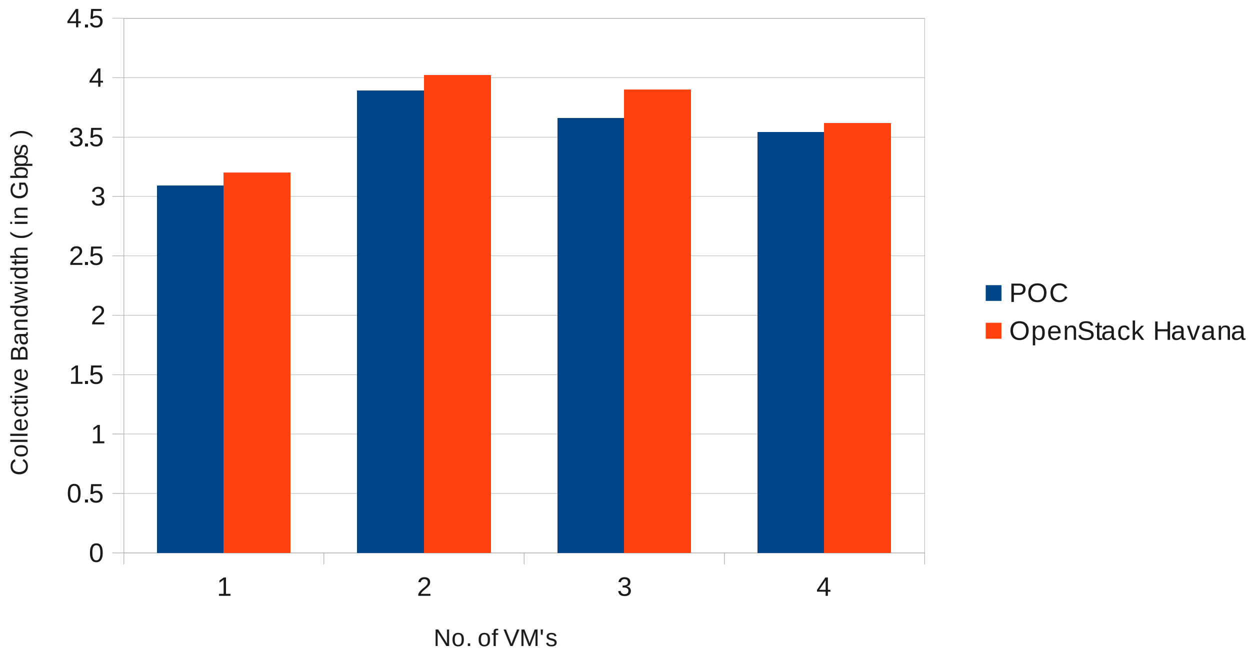
Use Case 1 – SNAT for single host
For this use case, we deployed multiple VM’s on same compute node and checked the collective bandwidth for various VM’s on a single node for SNAT



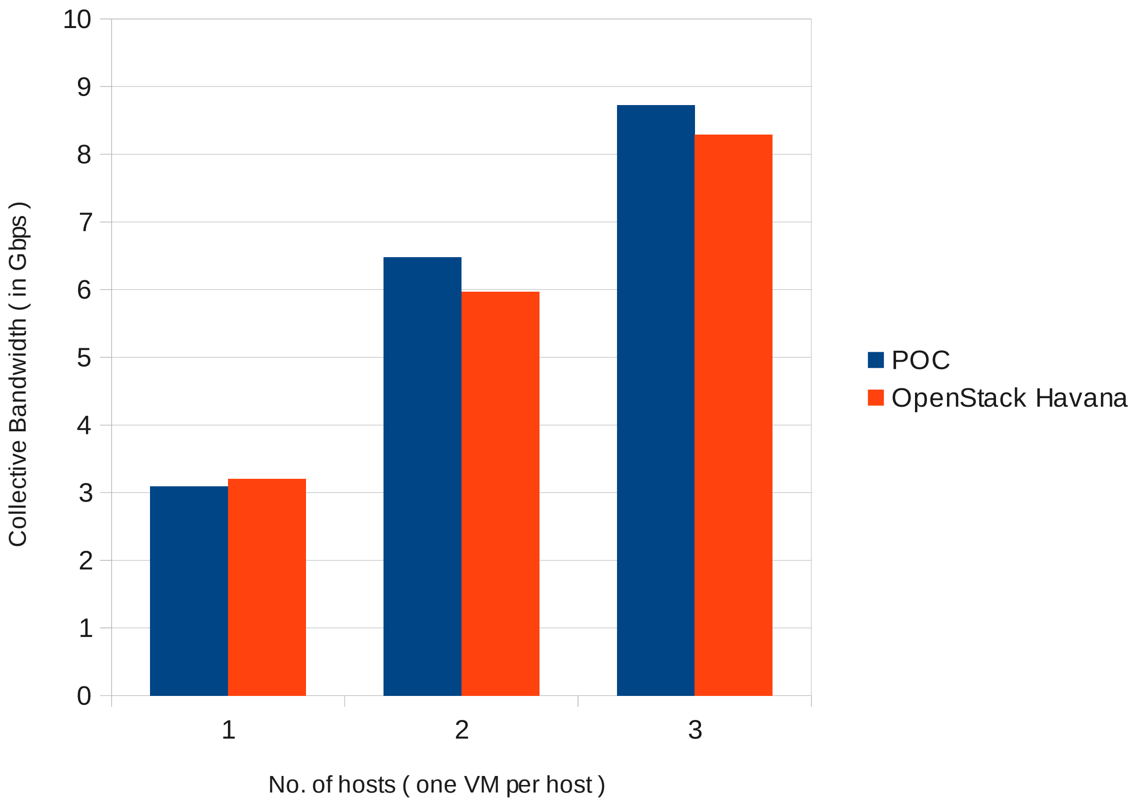
Use case 2- SNAT for Multiple Hosts
For this use case, we deployed multiple VM’s on multiple compute nodes and checked the collective bandwidth for various VM’s on multiple hosts for SNAT



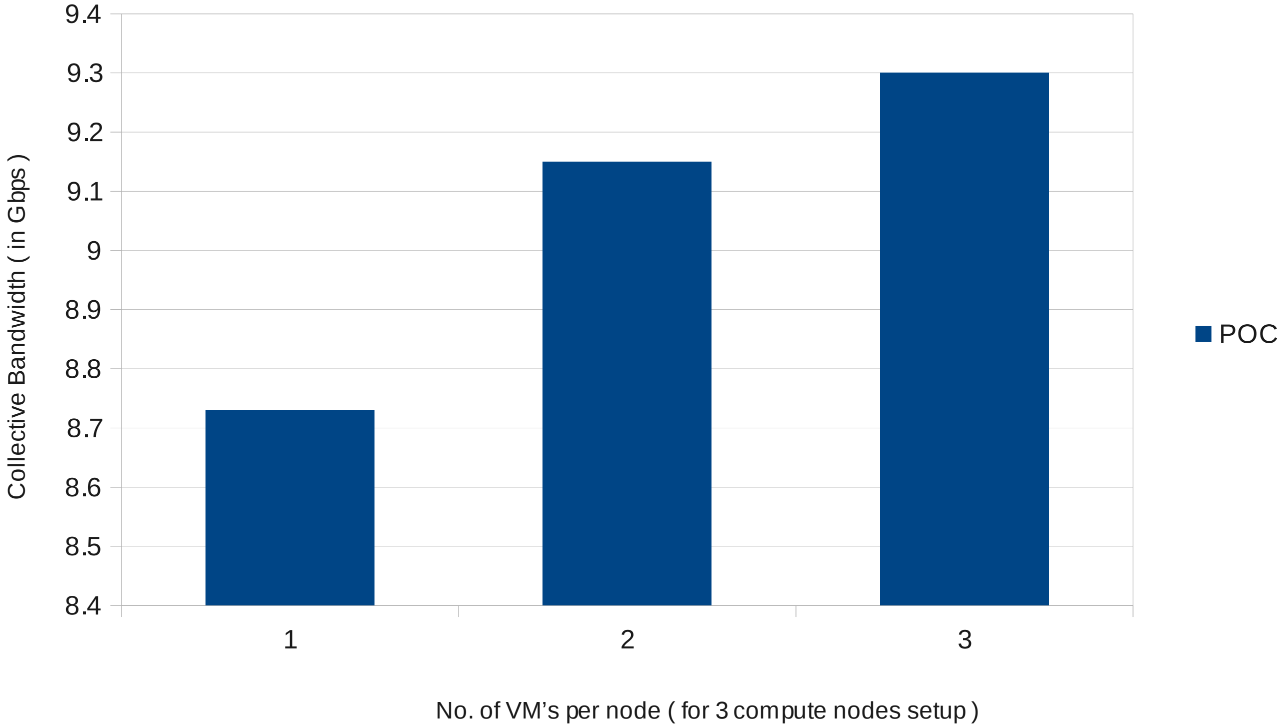
Use Case 3 – DNAT for single host
For this use case, we deployed multiple VM's on same compute node and checked the collective bandwidth for various VM's on a single host for DNAT



Use case 4- DNAT for Multiple Hosts
For this use case, we deployed multiple VM's on multiple compute nodes and checked the collective bandwidth for various VM's on multiple hosts for DNAT



Use Case 5- DNAT for Multiple Hosts (for POC only)



PREPARE SETUP FOR POC

Refer to the CookBook accompanying the document

REFERENCES:

- https://wiki.openstack.org/wiki/Neutron/ML2
- https://blueprints.launchpad.net/neutron/+spec/l2-population
- https://docs.google.com/document/d/1sUrvOQ9GIl9IWMGg3qbx2mX0DdXvMiyvCw2Lm6snaWQ/edit
- https://review.openstack.org/#/c/49227
- http://docs.openstack.org/
- http://git.openvswitch.org/cgi-bin/gitweb.cgi?=openvswitch;a=blob_plain;f=tutorial/Tutorial;hb=HEAD
- https://wiki.openstack.org/wiki/Neutron/DVR_L2_Agent
- https://wiki.openstack.org/wiki/Neutron/DVR
- https://blueprints.launchpad.net/neutron/+spec/neutron-ovs-dvr
- https://wiki.openstack.org/wiki/Neutron/DVR/HowTo
- https://docs.google.com/document/d/1iXMAyVMf42FTahExmGdYNGOBFyeA4e74sAO3pvr_RjA/edit

CREDITS

Rachappa B Goni
Advisory Engineer
Cloud Networking, GTS, IBM India Pvt. LTD
grachapp@in.ibm.com

Prashanth K Nageshappa
Senior Engineer, Master Inventor

Cloud Networking, GTS, IBM India Pvt. LTD
nprashan@in.ibm.com

Gowtham Narasimhaiah
Development Manager
Cloud Networking, GTS, IBM India Pvt. LTD
ngowtham@in.ibm.com

Vaidyanathan Gopalakrishnan
Operations Manager
Cloud Networking, GTS, IBM India Pvt. LTD
vaigopal@in.ibm.com