ADD L2POP SUPPORT FOR

FLOATINGIP RESOURCES

Layer 2 population (l2pop) mechanism driver implements the ML2 driver to improve open source plugins overlay implementations (VXLAN with Linux bridge and GRE/VXLAN with OVS)**[1]**. L2pop avoid the broadcast in mac learning and ARP resolution by prepopulate the bridge forwarding table**[2]**. However l2pop doesn’t support for floating ip resources and it will be better to add l2pop support for floating ip resources.

Problem Description

Figure-1 illustrates the scenarios for floating ip. The ip address(182.34.4.2) in port3 is the ip address in floating ip resources. When the client in the external network tries to access VM1, the destination ip(182.34.4.2) is replaced by the ip address(10.0.0.3) of the port1. Notice that the ip address of port1 is the ip address of VM1. When VM1 send packets back to client, the source ip(10.0.0.3) will be replaced by 182.34.4.2.



When users want to use floating ip, they should first require allocating a floating ip from floating ip pool or choose an pre allocated floating ip. Allocating floating ip will create a new port and also updates the new port. Creating port will never trigger l2pop but updating port will trigger l2pop on the condition that the port’s new status is ACTIVE or DOWN. However the status of the port for floating ip will always be N/A. Thus allocating floating ip won’t trigger l2pop.

After getting the available floating ip, users can require to associate the vm and the floating ip. Associating the ip address of the vm with the floating ip only updates the floating ip object and doesn’t update the port for floating ip. Thus associating the vm and the floating ip won’t trigger l2pop.



Based on the above analysis, the fdb of floating ip is not pre populated. In the following use case, arp request will be sent out to query the mac address related to the specific floating ip. Figure-2 illustrates an environment(DVR disabled) that there are two network nodes and two compute nodes.

Use Case 1

1. Tenant-1 creates Network-1 and tenant-2 creates Network-2.
2. Tenant-1 creates Subnet-1 belonging to Network-1 and Tenant-2 creates Subnet-2 belonging to Network-2.
3. Tenant-1 creates Router-1 and links the Router-1 to the external provider network. Tenant-2 creates Router-2 and links the Router-2 to the external provider network.
4. Tenant-1 links the Router-1 to the Subnet-1 and tenant-2 links Router-2 to the Subnet-2.
5. Tenant-1 creates VM-1 in Subnet-1 and tenant-2 create VM-2 in Subnet-2.
6. Tenant-1 creates Floatingip-1 and tenant-2 creates Floatingip-2.
7. Tenant-1 associates VM-1 with Floatingip-1 and tenant2 associates VM-2 with Floatingip-2.
8. VM-1 communicates VM-2 by floatingip. For example: VM-1 pings Floatingip-2 for the first time.

VM-1 has the ip address “10.0.0.1” which is the ip address of Port1 and VM-2 has the ip address “10.0.0.2” which is the ip address of Port2. Floatingip-1 has the ip address “182.34.4.5” which is the ip address of Port5 and floatingip-2 has the ip address “182.34.4.6” which is the ip address of Port6. The mac address of Router-1 gateway is “fa:16:3e:1b:ee:2b” and the mac address of Router-2 gateway is “fa:16:3e:ab:cf:34”.

VM-1 and VM-2 reside on different networks and communicate via their respective floating ip addresses. When VM-1 pings Floatingip-2 for the first time, it needs to know the mac address for Floatingip-2. Thus arp request is sent out. And the mac address(“fa:16:3e:ab:cf:34”) of Router-2 gateway will be answered. If the fdb is pre-populated, the arp request can be avoided. For non-DVR use case, (host\_ip, router\_gateway\_mac\_address, floating\_ip\_address) should be pre-populated.

The figure-3 illustrates an environment(DVR enabled). The use case is listed below:

Use Case 2

1. Tenant-1 creates Network-1 and tenant-2 creates Network-2.
2. Tenant-1 creates Subnet-1 belonging to Network-1 and Tenant-2 creates Subnet-2 belonging to Network-2.
3. Tenant-1 creates a distributed virtual router DVR-1 and Router-1 is the distributed router on compute-1. Tenant-2 creates a distributed virtual router DVR-2 and Router-2 is the distributed router on compute-2.
4. Tenant-1 links the DVR-1 to the external provider network. Tenant-2 links the DVR-2 to the external provider network.
5. Tenant-1 links the DVR-1 to the Subnet-1 and tenant-2 links DVR-2 to the Subnet-2.
6. Tenant-1 creates VM-1 in Subnet-1 and tenant-2 create VM-2 in Subnet-2.
7. Tenant-1 creates Floatingip-1 and tenant-2 creates Floatingip-2.
8. Tenant-1 associates VM-1 with Floatingip-1 and tenant2 associates VM-2 with Floatingip-2.
9. VM-1 communicates VM-2 by floatingip. For example: VM-1 pings Floatingip-2 for the first time.



VM-1 has the ip address “10.0.0.1” which is the ip address of Port1 and VM-2 has the ip address “10.0.0.2” which is the ip address of Port2. Floatingip-1 has the ip address “182.34.4.5” which is the ip address of Port5 and floatingip-2 has the ip address “182.34.4.6” which is the ip address of Port6. When a floating IP is attached to a VM, the L3 agent will create a FIP namespace (If one does not already exist) for the external network that the FIP belongs to. After the step 8, FIP-1 and FIP-2 will be created. Port9 is FIP-1 floatingip agent gateway and Port10 is FIP-2 floatingip agent gateway. The mac address of Port9 is “fa:16:3e:e9:87:24” and the mac address of Port10 is “fa:16:3e:e7:86:db”.

VM-1 and VM-2 reside on different networks and communicate via their respective floating ip addresses. When VM-1 pings Floatingip-2 for the first time, it needs to know the mac address for Floatingip-2. Thus arp request is sent out. And the mac address(“fa:16:3e:e7:86:db”) of FIP-2 floatingip agent gateway will be answered. If the fdb is prepopulated, the arp request can be avoided. For DVR use case, (host\_ip, floatingip\_agent\_gateway\_mac\_address, floating\_ip\_address) should be prepopulated.

Proposed Change

The idea is that advertising the fdbs for floating ip when the FIP status changes to "ACTIVE" and withdraw the fdbs for floating ip whenever the status is set to "DOWN" or the resource is deleted or disassociated.

Function \_notify\_fip\_status will be added to send an event after updating floating ips.  Function \_delete\_floatingip should be modified to use \_notify\_fip\_status to notify the floating ip status. Function \_update\_fip\_assoc is used by function \_create\_floatingip and \_update\_floatingip. Both functions will send events after updating floating ip with the result from \_update\_fip\_assoc. Thus we can just insert the status of floating ip into the result of \_update\_fip\_assoc. Function disassociate\_floatingips already sends event out after updating floating ip. However after disassociating floating ips, the router\_id and fixed\_port\_id both are None. Thus need to insert floating ip status, last\_known\_router\_id and last\_fixed\_port\_id into the event.

A class L3RouterL2PopMixin should be added to process the event sent after updating floating ip. This class should subscribe the event and register the callback to l2pop mechanism driver to extend the l2pop fdbs(Depends on RFE: [https://bugs.launchpad.net/neutron/+bug/1793653](https://bugs.launchpad.net/neutron/%2Bbug/1793653)). Thus Callback l3\_fdb\_extend\_func should be added. Function handle\_fip\_status\_update should be added to process the event and send the l2pop fdbs out. For non-DVR, (host\_ip, router\_gateway\_mac\_address, floating\_ip\_address) should be pre-populated. For DVR, (host\_ip, floatingip\_agent\_gateway\_mac\_address, floating\_ip\_address) should be pre-populated. What’s more, some other functions used by the functions described above should be added.

All changes can be viewed through the link below:

<https://review.openstack.org/#/c/611261/>

https://review.openstack.org/#/c/611284/

Data Model Impact

None

REST API Impact

None

Command Line Client Impact

None

Other Impact

None

Other Deployer Impact

None

Performance Impact

Performance testing should be conducted to see what is the overhead of adding more information to fdb.

Implementation

Assignee(s)

Work Items

* Add function \_notify\_fip\_status to neutron/db/l3\_db.py and modify some existing functions in l3\_db.py.
* Add class L3RouterL2PopMixin to neutron/services/l3\_router/service\_providers/l2pop.py
* Add related tests.

Dependencies

Need the RFE “Enable other subprojects to extend l2pop fdb information” merged  in openstack. This RFE enables other subprojects to extend l2pop fdb information. We need to add floating ip related information to l2pop fdb. Thus we need to use the ability provided by the RFE. The link for RFE is below:

https://bugs.launchpad.net/neutron/+bug/1793653

Testing

Unit tests are necessary.

Documentation Impact

None.

References

**[1]** <https://github.com/openstack/neutron/tree/master/neutron/plugins/ml2/drivers/l2pop>

**[2]** <https://wiki.openstack.org/wiki/L2population_blueprint>