OpenStack Network Plugins Development Proposal

By:
Tom Burke
Shen Han
Sigurdur Thorvaldsson
Hanjie Zhang

1. Vision and Goals Of The Project:

The Openstack Neutron Cisco ASR1K plugin main goal is to integrate OpenStack Neutron with Cisco's ASR1K Router. Our team will identify and potentially add support for existing Neutron features not yet supported by the plugin.High-Level Goals of the Openstack Neutron Cisco ASR1K plugin and our team include:

- Ability to add custom "out of band" config from OpenStack Neutron to the ASR1K router
- Potential Firewall rule translation from OpenStack Neutron to the ASR1K router
- Fully automated unit and functional tests for "out of band config" and firewall support

The Openstack ASR1K plugin implements openstack tenant router instances created by openstack neutron API calls. The tenant routers are implemented in the physical ASR1K's as virtual router forwarding (VRF) instances attached to tenant networks via VLAN subinterfaces.

Users/Persons Of The Project:

The Openstack Neutron Cisco ASR1K plugin is currently being used many major customers of Cisco. We will understand how customers are using the plugin and will review enhancement possibilities. The out-of-band config is specifically targeted at openstack admin users. This plugin will not be used by individuals who do not have access to a Cisco ASR1K router.

2. Scope and Features Of The Project:

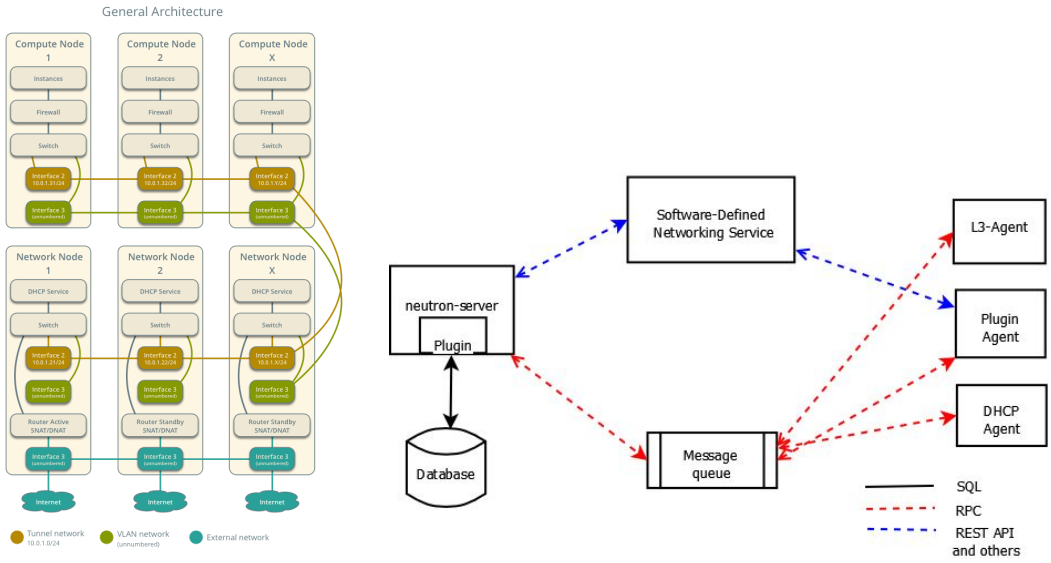

Figure 1 the general architecture of the openstack network[1](left) networking flow diagram of the OpenStack Networking[2](right)

What has already been done:

To understand the scope of this project it is best to start by understanding what has already been implemented on the Openstack Neutron Cisco ASR1K plugin. We will use Figure 1 to give a clear description of what has already been implemented as our background. Neutron's ML2 plugins take level 2 configurations from Neutron and implement them in the appropriate switches. In layer 3 OpenStack stores all information about the subnet (default gateway IP and NATed IPs) in the virtual router. The plugin currently can provide an environment to terminate the user facing API. This information can then be translated for ASR1K configuration to add additional network directing rules. So as of right now Cisco's plugin can integrate a subnet with dynamic NAT from OpenStack Neutron to the ASR1K.

What we'll work on

The features we will work on are listed as below.

There are two main goals of the OpenStack Network Plugins Development Project.

1. Out of Band Config:

[1] "Scenario: High Availability using Distributed Virtual Routing (DVR)", Openstack.org. See more at http://docs.openstack.org/liberty/networking-guide/scenario-dvr-ovs.html

[2] "Networking architecture", Openstack.org. See more at http://docs.openstack.org/security-guide/networking/architecture.html

The first is to add "out of band config" capabilities between the interface between OpenStack Neutron and the ASR1K Plugin. "Out of band config" is adding configurations that are not supported on OpenStack Neutron and integrating them with the ASR1K. For example, the ASR1K has the capability to turn on and off Netflow, but Neutron currently has no setup. Netflow is a network protocol developed by Cisco for collecting IP traffic information and monitoring network traffic. By analyzing flow data, a picture of network traffic flow and volume can be built. Our goal in this specific case be to add a configuration file on Neutron that would allow the cloud admin to set these configurations, which could then be translated and applied to the ASR1K. Another example of an "out of band config" could be SNMP pertaining to the per-VRF SNMP configuration, which is important because it is a popular protocol for network management. It is used for collecting information from, and configuring, network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP) network. More specifically SNMP pertaining to the per-VRF SNMP configuration can be used to allow a per-openstack tenant SNMP notification receiver to get ASR1K notifications confined to the context of an openstack tenant router.

Openstack cloud-administrators would like the capability to have the ASR1K plugin to create/delete some free-form configuration on their behalf during the neutron tenant router lifecycle events. This capability should rely on configuration scoping in the ASR1K. Example config scopes:

- Global config scope
- sub-interface config scope: specifically the interfaces created/deleted by the ASR1K plugin to attach to openstack tenant networks
- VRFs mapping to tenant router instances

2. Firewall support:

The second vision is more of an optional one from our mentors. The goal would be to set up firewall support using Neutron. We would need to add a feature to the Openstack Neutron Cisco ASR1K plugin to take firewall rules from Neutron and translate them into a form that the ASR1K can use. The plugin will do all of the following:

- Terminate the user facing API
- Digest attributes coming in from the API
- Perform Validation
- Populate DB
- RPC to Cisco Config Agent

What we won't work on

We will not be working on setting up subnets on the ASR1K via Neutron's virtual router because this has already been accomplished by Cisco. We will not work on any "out of band" configurations that Cisco clients are not interested in.

3. Solution Concept

Global Architectural Structure Of the Project:

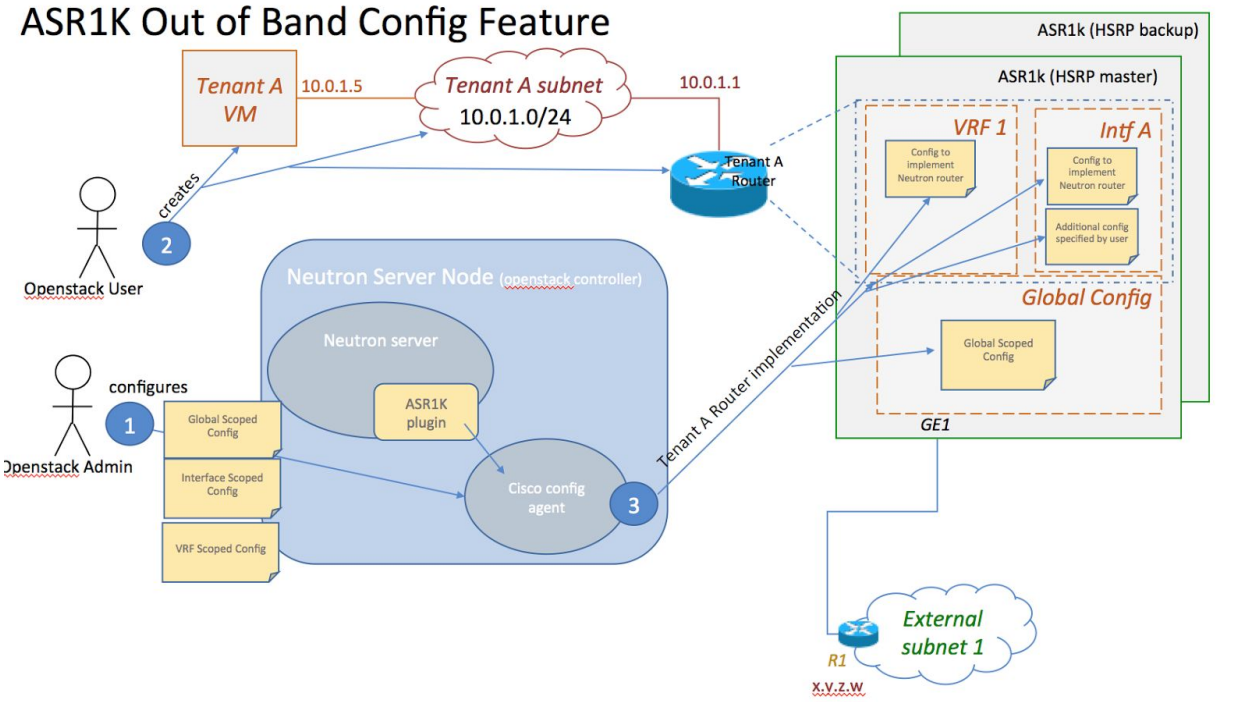

Figure 2 ASR1K out of band config feature[3]

Figure 2 shows the Out of Band Config Feature we will aim to implement. The tentative steps that we will take include:

1. Admin configures the ASR1K plugin/config agent with additional config (template) to apply to the ASR1K along with the config to implement any tenant routers. (config is either via config agent config files OR via neutron API extension.
2. User creates a tenant A router
3. ASR1K plugin & config agent implements the tenant router via VRF1, Intf A, and global config in the ASR; ASR1K plugin & config agent apply additional config defined by user

[3] "ASR1k Out-of-Band Config Feature", Timothy Swanson, page 7.

The architecture structure includes:

- A config file that includes the out-of-band configurations for all scopes, like global config scope. The detailed configurations include turning on/off the netflow and limiting the traffic and so on.
- An interface where user can set up the out-of-band configurations.
- Policy that checks that the out-of-band config supplied is valid, for example ensuring that an IPv4 address is of an appropriate length/format.

Firewall functionality is architecturally another service bubble on the network-node that the plugin could set up the ASR1K to implement.

The main structure includes:

- Firewall rules: The actual rules of the firewall
- Firewall policy: Collection of firewall rules
- Firewall groups: A binding of ingress & egress Firewall Policies to ports where they are applied.
- Cisco config agent: Maps canonical representation for dict with firewall rules to proper VRF, interface, generates ordered ACL rules corresponding to Firewall Rules
- Driver layer:Talks to the ASR1K using netconf, pushes the config to the ASR1K

Design Implications and Discussion

- The config file is the library where out-of-band configurations are kept, the existence of such a library tremendously ease the configuration process.
- The out-of-band configuration policy: It is acting like a guard to check the configuration is valid before it will be implemented on the ASR1K. If such a guard doesn't exist, ASR1K router may receive the illogical configuration and then either replies the mistake or silently dismisses the configuration. The former may make the configuration process longer as the check process is done by the ASR1K and the latter could fail the configuration directly.
- Firewall config agent: Kinds of rules for VRF, interface are set to implement with the router. This agent can automatically map different rules to different components.

4. Acceptance Criteria

Minimum acceptance criteria is to add or edit some existing features in the current Openstack Neutron Cisco ASR1K plugin to provide additional functionality or support. These goals may include:

- Firewall support:
    - Terminates the user facing API
    - Digests the attributes coming from API
    - RPC to the agent.

    - Test if ASR1k gets configured per the API flows.
- Out of band configuration:
    - A config file that could be applied to the Cisco config agent to implement the tenants' routers.
    - Test if ASR1k gets configured per the API flows.
- Existing ASR1k configuration regression gets passed with no errors.
- A successful demonstration to the class.
- All changes are committed to the common project version control repository, such as openstack/networking-cisco github repo.

In addition, all the defects existing in our plugin should be filed per the project practice.

5. Release Planning

5.1 Setup, workflows and tools (spike) (due 02/23, first sprint, two weeks.)
In this sprint we will get familiar with OpenStack, workflows and some tools we will be using. The points we will be working on are:

- Setup VM
- Devstack
- Try out Horizon & CLI
- Explore common workflows
- Create Router
- Create Network, subnet
- Attach interface to Router
- PyCharm
    - Install
    - Open neutron as project
    - Connect to neutron DB and inspect tables and their content

- Freshen up on Ethernet, VLAN, routing, HTTP protocol, REST, tcpdump, Linux network namespaces, openvswitch
- Understand Plugin, Agent, Driver concepts
- Create VM
- Security Groups
- Attach Floating IP
- Traffic from Inside to Outside and Outside to inside
- Understand Basic ideas behind NAT
- Adding breakpoints in the neutron server and Agent and follow the code for one of the workflows above.
- Using Pycharm
    - Run neutron server in Pycharm

    - Set breakpoints and track processing of REST api call to create router
    - Remote debugging using Pycharm

- Understand the packet path from a VM to
    - Another VM on the same subnet
    - A VM on a different subnet
    - Destination outside the OpenStack cloud

- Understand the neutron extension/plugin/agent layers
- Understand Data models and look at mysql tables

5.2 Bringing up an ASR1K (due 03/16, second sprint, two weeks).
In this sprint we will implement what we did in sprint 1 on an ASR1K. The points we will be working on are:

- devstack environment to bring up an ASR1K
- Logging in to the ASR and looking at basic config
- Use IOS XE CLI
- Repeat common workflows, check impact on ASR config
- Understand the ASR1K Plugin and Config Agent
- Differences from the Community Implementation
- Follow the code for a simple workflow

5.3 Design for feature enhancement (due 03/30, third sprint, two weeks.)
The points we will be working on are:

- Out of Band configuration
- Firewall
- Understand and define the changes for enhancement
- Design outline document covering the different pieces

5.4 Implementation (due 04/27, fourth and fifth sprint, four weeks)
In this sprint we will tie up any loose ends and do some unit testing if we have time.