SDNVPN Testing

- Current test cases
  - Overview
  - Functest - scenario specific tests
    - Test Case 1 - VPN provides connectivity between subnets
    - Test Case 2 - tenant separation
    - Test Case 3 - Data Center Gateway integration
    - Test Case 4 - VPN provides connectivity between subnets using router association
    - Test Case 7 - Network associate a subnet with a router attached to a VPN and verify floating IP functionality (disabled)
    - Test Case 8 - Router associate a subnet with a router attached to a VPN and verify floating IP functionality
    - Test Case 9 - Check fail mode in OVS br-int interfaces
    - Test Case 10 - Check the communication between a group of VMs
    - Test Case 11: test Opendaylight resync and group_add_mod feature mechanisms
  - Yardstick
  - ODL VPN Service tests
- Testing Roadmap
  - Target coverage for BGPVPN tempest tests
  - Use case testing backlog
    - Overview
    - Test case: Inter-Data Center connectivity
    - Router association
    - Static routes
    - Port association
  - Functest
  - Yardstick

Test compliance template for Dovetail

- General test case information
  1. Provide a high level description of the test area, main features being tested, and answer why it is in scope for Dovetail Danube. If there is a subset that is in scope, describe the subset.
  2. Describe how the features are tested, and justify why it is appropriate methodology for compliance testing. If modification or enhancement are needed, please describe the work needed in Dovetail.
  3. Describe how the automated test cases are implemented, and why it is appropriate for compliance testing. If modification or enhancement are needed, please describe the work needed in Dovetail.
  4. Describe test results, pass/fail criteria
  5. Describe the conditions that the SUT must be in for the test cases to run. Are the test cases executable in a limited subset of OPNFV Scenarios only? If so, which subset have the pre-conditions met to pass?
  6. Provide a link of pass test run results / examples, if any.
  7. Any other important information or comments

- Specific test case requirement / criteria
  1. If there is an industry standard, the test cases should follow the standard. Please provide references of the standards. If the existing standards are not followed, please provide justifications. An exception is required.
  2. If a compliance test suite exists for the upstream component being tested, the upstream test suite should be used as a baseline. Please describe the baseline upstream suite. If this is not followed, or a subset is excluded or modified from the upstream, please provide justifications. An exception is required.
  3. Test cases must be able to pass at least some OPNFV reference deployment or deployments, as deployed according to one or more scenarios in Danube. Please describe the scenarios that the test cases can pass (assuming the scenarios are correctly deployed - i.e. if it is due to errors of the SUT software, we don't disqualify a test case for that).
  4. The features being tested must not rely on unmerged patches in the upstream projects. In other words, unmerged patches are considered not mature enough for Dovetail suite. Please list the upstream projects and affirm this is the case to the relevant upstream projects.
  5. The features being tested must be in scope of CVP during the Danube cycle. Please affirm that this is the case for the proposed set of test cases.
  6. Any other important information or comments.

The aim is to run the same suite of tests for all supported backends, i.e. within the CI pipeline a separate Jenkins job is needed for each deployment option (deploy the option, run the common tests).

Current test cases

Overview

We currently have the following SDNVPN specific testing (the non-feature-specific test suites in Functest and Yardstick are being run against our scenarios in CI as well):

- BGPVPN Tempest test cases
  - Create BGPVPN passes
  - Create BGPVPN as non-admin fails
  - Delete BGPVPN as non-admin fails
- Functest (see details below)
  - Test case 1: VPN provides connectivity between subnets. Different Neutron subnets cannot reach each other unless they are connected to the same Router. Connecting two subnets to the same VPN is an alternative means to achieve inter-subnet connectivity.
- Test Case 2: tenant separation. VPNs enable using the same IP address ranges in different VPNs, which is an important feature for tenant separation in the DC. This test verifies if the correct VM is reached under a given IP address when IP addresses are used multiple times in the same DC.

There is a robot test suite for ODL VPN Service but it is not used in OPNFV currently (see below for an overview).

**Functest - scenario specific tests**

In the following we describe the tests we have added to Functest in order to verify correct function of our feature.

2 compute nodes Node1 and Node2 are used during the tests.

**Common test setup procedure:**

- tbd

**Common test teardown procedure:**

- Functest framework removes the VMs after the test execution
- tbd

**Test Case 1 - VPN provides connectivity between subnets**

**Name:** VPN connecting Neutron networks and subnets

**Description:** VPNs provide connectivity across Neutron networks and subnets if configured accordingly.

**Test setup procedure:**

- set up VM1 and VM2 on Node1 and VM3 on Node2, all having ports in the same Neutron Network N1 and all having 10.10.10/24 addresses (this subnet is denoted SN1 in the following)
- Set up VM4 on Node1 and VM5 on Node2, both having ports in Neutron Network N2 and having 10.10.11/24 addresses (this subnet is denoted SN2 in the following)

**Test execution:**

- Create VPN1 with eRT<>iRT (so that connected subnets should not reach each other) and associate SN1 to it
- Ping from VM1 to VM2 should work
- Ping from VM1 to VM3 should not work
- Associate SN2 to VPN1
- Ping from VM4 to VM5 should work
- Ping from VM1 to VM4 should not work (disabled until isolation fixed upstream)
- Associate SN2 to VPN1
- Ping from VM1 to VM5 should work
- Ping from VM1 to VM5 should work

**Jira task in Functest for the Test 1:** [https://jira.opnfv.org/browse/YARDSTICK-185](https://jira.opnfv.org/browse/YARDSTICK-185)
Test Case 2 - tenant separation

Name: Using VPNs for tenant separation
Description: Using VPNs to isolate tenants so that overlapping IP address ranges can be used

Test setup procedure:

- set up VM1 and VM2 on Node1 and VM3 on Node2, all having ports in the same Neutron Network N1.
- VM1 and VM2 have IP addresses in a subnet SN1 with range 10.10.10/24
  - VM1: 10.10.10.11, running an HTTP server which returns "I am VM1" for any HTTP request (or something else than an HTTP server)
  - VM2: 10.10.10.12, running an HTTP server which returns "I am VM2" for any HTTP request
- VM3 has an IP address in a subnet SN2 with range 10.10.11/24
  - VM3: 10.10.11.13, running an HTTP server which returns "I am VM3" for any HTTP request
- Set up VM4 on Node1 and VM5 on Node2, both having ports in Neutron Network N2
- VM4 has an address in a subnet SN1b with range 10.10.10/24
  - VM4: 10.10.10.12 (the same as VM2), running an HTTP server which returns "I am VM4" for any HTTP request
- VM5 has an address in a subnet SN2b with range 10.10.11/24
  - VM5: 10.10.11.13 (the same as VM3), running an HTTP server which returns "I am VM5" for any HTTP request

Test execution:

- Create VPN 1 with iRT=eRT=RT1 and associate N1 to it
  - HTTP from VM1 to VM2 and VM3 should work
    - It returns "I am VM2" and "I am VM3" respectively
  - HTTP from VM1 to VM4 and VM5 should not work
    - It never returns "I am VM4" or "I am VM5"
- Create VPN2 with iRT=eRT=RT2 and associate N2 to it
  - HTTP from VM4 to VM5 should work
    - It returns "I am VM5"
  - HTTP from VM4 to VM1 and VM3 should not work
    - It never returns "I am VM1" or "I am VM3"

Jira task in Yardstick for the Test2: https://jira.opnfv.org/browse/YARDSTICK-192

Test Case 3 - Data Center Gateway integration

Name: Data Center Gateway integration
Description: Investigate the peering functionality of BGP protocol, using a Zrpcd/Quagga router and OpenDaylight Controller

Test setup procedure:

- Search in the pool of nodes and find one Compute node and one Controller nodes, that have OpenDaylight controller running
- Start an instance using ubuntu-16.04-server-cloudimg-amd64-disk1.img image and in it run the Quagga setup script
- Start bgp router in the Controller node, using odl:configure-bgp

Test execution:

- Set up a Quagga instance in a nova compute node
- Start a BGP router with OpenDaylight in a controller node
- Add the Quagga running in the instance as a neighbor
- Check that bgpd is running
- Verify that the OpenDaylight and gateway Quagga peer each other
- Start an instance in a second nova compute node and connect it with a new network, (Network 3-3).
- Create a bgpvpn (include parameters route-distinguisher and route-targets) and associate it with the network created
- Define the same route-distinguisher and route-targets on the simulated quagga side
- Check that the routes from the Network 3-3 are advertised towards simulated Quagga VM

Note: The file containing quagga (zrpcd) package and its dependencies can be downloaded from here:

rpm packages: quagga-4.tar.gz (cksum)

deb packages: quagga-ubuntu.tar.gz (cksum)

(Danube version quagga-3.tar.gz- rpm) quagga-3.tar.gz- rpm)

Test Case 4 - VPN provides connectivity between subnets using router association

Functest: variant of Test Case 1. Set up a Router R1 with one connected network/subnet N1/S1. Set up a second network N2. Create VPN1 and associate Router R1 and Network N2 to it. Hosts from N2 should be able to reach hosts in N1.

Name: VPN connecting Neutron networks and subnets using router association
Description: VPNs provide connectivity across Neutron networks and subnets if configured accordingly.

Test setup procedure:

- Set up VM1 and VM2 on Node1 and VM3 on Node2, all having ports in the same Neutron Network N1 and all having 10.10.10/24 addresses (this subnet is denoted SN1 in the following). N1/SN1 are connected to router R1.
- Set up VM4 on Node1 and VM5 on Node2, both having ports in Neutron Network N2 and having 10.10.11/24 addresses (this subnet is denoted SN2 in the following)

Test execution:

1. Create VPN1 with eRT<>iRT (so that connected subnets should not reach each other) and associate R1 to it
   - Ping from VM1 to VM2 should work
   - Ping from VM1 to VM3 should work
   - Ping from VM1 to VM4 should not work
2. Associate SN2 to VPN1
   - Ping from VM4 to VM5 should work
   - Ping from VM1 to VM4 should not work
   - Ping from VM1 to VM5 should not work
3. Change VPN1 so that iRT=eRT
   - Ping from VM1 to VM4 should work
   - Ping from VM1 to VM5 should work

Jira task: https://jira.opnfv.org/browse/SDNVPN-49
Test Case 7 - Network associate a subnet with a router attached to a VPN and verify floating IP functionality (disabled)

A test for [https://bugs.opendaylight.org/show_bug.cgi?id=6962](https://bugs.opendaylight.org/show_bug.cgi?id=6962)

Setup procedure:
- Create VM1 in a subnet with a router attached.
- Create VM2 in a different subnet with another router attached.
- Network associate them to a VPN with iRT=eRT
- Ping from VM1 to VM2 should work
- Assign a floating IP to VM1
- Pinging the floating IP should work

Test Case 8 - Router associate a subnet with a router attached to a VPN and verify floating IP functionality

Setup procedure:
- Create VM1 in a subnet with a router which is connected with the gateway
- Create VM2 in a different subnet without a router attached.
- Assoc the two networks in a VPN iRT=eRT
- One with router assoc, other with net assoc
- Try to ping from one VM to the other
- Assign a floating IP to the VM in the router assoc network
- Ping it

Test Case 9 - Check fail mode in OVS br-int interfaces

This testcase checks if the fail mode is always "secure".

To accomplish it, a check is performed on all OVS br-int interfaces, for all OpenStack nodes.

The testcase is considered as successful if all OVS br-int interfaces have fail_mode=secure

Test Case 10 - Check the communication between a group of VMs

This testcase investigates if communication between a group of VMs is interrupted upon deletion and creation of VMs inside this group.

Test case flow:
- Create 3 VMs: VM_1 on compute 1, VM_2 on compute 1, VM_3 on compute 2. All vms ping each other.
- VM_2 is deleted. Traffic is still flying between VM_1 and VM_3.
- A new VM, VM_4 is added to compute 1. Traffic is not interrupted and VM_4 can be reached as well.

Testcase 11: test Opendaylight resync and group_add_mod feature mechanisms

This is testcase to test Opendaylight resync and group_add_mod feature functionalities
Sub-testcase 11-1:

- Create and start 2 VMs, connected to a common Network. New groups should appear in OVS dump
- OVS disconnects and the VMs and the networks are cleaned. The new groups are still in the OVS dump, cause OVS is not connected anymore, so it is not notified that the groups are deleted
- OVS re-connects. The new groups should be deleted, as Opendaylight has to resync the groups totally and should remove the groups since VMS are deleted.

Sub-testcase 11-2:

- Create and start 2 VMs, connected to a common Network. New groups should appear in OVS dump
- OVS disconnects. The new groups are still in the OVS dump, cause OVS is not connected anymore, so it is not notified that the groups are deleted
- OVS re-connects. The new groups should be still there, as the topology remains. Opendaylight Carbon' group_add_mod mechanism should handle the already existing group.

Yardstick

The following tests are enabled in yardstick:

- opnfv_yardstick_tc002.yaml - Network Latency
- opnfv_yardstick_tc005.yaml - Storage Performance
- opnfv_yardstick_tc043.yaml - Network Latency Between NFVI Nodes

ODL VPN Service tests

Name: ODL VPN Service API tests
Description: Robot Framework tests for ODL VPN service using ODL REST API.

The ODL integration-test repository is already cloned by Functest. We can however currently not run these tests as-is since they use Mininet to set up a test topology and we have a real topology in our setup. The current Functest Test Case 1 covers the scope of this node level testing.

1. Create VPN instance and check command return code
2. Check if VPN instance is present
3. Create IETF VM interface and check return code
4. Verify IETF VM interface
5. Create VPN interface for IETF interface
6. Verify VPN interface
7. Verify FIB entry after create
8. Delete VM VPN interface
9. Verify after deleting VM VPN interface
10. Delete VPN instance
11. Verify after deleting VPN instance
12. Delete VM IETF interface
13. Verify after deleting VM IETF interface
14. Verify FIB entry after delete

Testing Roadmap

Target coverage for BGPVPN tempest tests

BGPVPN Tempest tests (see https://github.com/openstack/networking-bgpvpn/tree/master/networking_bgpvpn_tempest) are for simple API verification, i.e. the API is invoked and the test checks if the returned result is as expected. This is the target list of API test cases (the list is constantly updated and implemented tests are moved to the Current Tests section above).

1. Create BGPVPN
2. Create BGPVPN with malformed route target (e.g. ASN:NN) should fail.
3. Create BGPVPN with invalid route target (e.g. 65536:0) should fail.
4. Getting the VPN list works without producing an error.
5. Updating an existing BGPVPN works.
6. Displaying parameters of an existing BGPVPN works.
7. Deleting a BGPVPN works.
8. Associating an existing BGPVPN with a Neutron network works
9. Getting the associated Neutron network works.
10. Deleting the network association works.
11. Associating an existing BGPVPN with a Neutron Router works
12. Getting the associated Neutron router works.
13. Deleting the router association works

Use case testing backlog

Complex test cases representing real system-level use cases are used to verify end-to-end functionality.
Overview

1. Test Case 3: Data Center Gateway integration: check if routes are correctly exchanged between SDN controller and datacenter gateway (extend the existing case)
2. Test Case 4: Inter-DC communication: Check if communication between two data centers connected through an MPLS backbone works.

Test case: Inter-Data Center connectivity

Test if connectivity between two physically separated data centers over the Internet (with DC-GW in between) works. To be clarified if and how this can be integrated with CI.

Router association

Tempest tests for API verification and functional OPNFV-level tests in Functest.

Static routes

As soon as static route is supported in BGPVPN, we can support some additional scenarios such as Hub'n'Spoke.

Port association

xyz.

Functest

Within Functest the basic functionality of the ODL VPN Service is verified. Functest calls the BGPVPN Temptest tests and defined additional tests directly in the Functest framework.

Yardstick

tbd

Test compliance template for Dovetail

General test case information

1. Provide a high level description of the test area, main features being tested, and answer why it is in scope for Dovetail Danube. If there is a subset that is in scope, describe the subset.

Scope of the OPNFV BGPVPN feature is to provide MPLS / Multiprotocol BGP based layer 2/3 VPN services in the OPNFV platform. The C&C committee has not yet defined the scope for Danube testing (the document linked in the Dovetail wiki refers to a not yet existing addendum) so it is not possible to evaluate if the BGPVPN feature is in scope.

In the Danube release the main features being tested are:

- Test case 1: Connectivity between Neutron subnets through association of Neutron Networks to VPNs
- Test case 2: Separation of tenant networks through association to different VPNs
- Test case 3: Data center gateway integration through BGP peering
- Test case 4: VPN provides connectivity between subnets using association of Neutron Router to VPNs
- Test Case 8: associate Neutron Router with an attached subnet to a VPN and verify reachability of the Floating IP

2. Describe how the features are tested, and justify why it is appropriate methodology for compliance testing. If modification or enhancement are needed, please describe the work needed in Dovetail.

The features are tested through the BGPVPN API (a Neutron service plugin), which implements the basic BGPVPN OpenStack Blueprint https://review.openstack.org/#/c/177740/. The blueprint represents the agreed way of integrating BGPVPN in OpenStack, and it complies with the relevant IETF standards, mainly RFC4364 (http://tools.ietf.org/html/rfc4364) and RFC7432 (http://tools.ietf.org/html/rfc7432). The tests cover fundamental BGPVPN capabilities and ensure that basic use cases can be realized using the OPNFV reference stack (i.e. the bgpvpn deployment scenarios).

3. Describe how the automated test cases are implemented, and why it is appropriate for compliance testing. If modification or enhancement are needed, please describe the work needed in Dovetail.

The tests are implemented using the OPNFV functest framework and invoke the BGPVPN service plugin API, including the BGPVPN-related tests integrated in upstream Tempest. The Heat and Horizon extensions are currently not used for testing.

Comment: the requirements towards the implementation of the test in the sense of "appropriateness for compliance testing" is not clear. What should be covered here that is not already covered under previous question?
4. Describe test results, pass/fail criteria

The pass/fail criteria for the currently implemented tests is if the desired connectivity (which in some cases is the absence of connectivity, i.e. VMs should not be able to reach each other if the VPN configuration is supposed to prevent that) is present. This is verified through ping tests.

5. Describe the conditions that the SUT must be in for the test cases to run. Are the test cases executable in a limited subset of OPNFV Scenarios only? If so, which subset have the pre-conditions met to pass?

The BGPVPN tests require the deployment of one of the BGPVPN deployment scenarios, which are currently supported by the Fuel and Apex installers. Other than that there are no requirements or assumptions on SUT state made by the tests, i.e. they can run any time regardless of previous activity on the SUT.

6. Provide a link of pass test run results / examples, if any.

See for example here.

7. Any other important information or comments

None.

Specific test case requirement / criteria

1. If there is an industry standard, the test cases should follow the standard. Please provide references of the standards. If the existing standards are not followed, please provide justifications. An exception is required.

The de-facto industry standard for BGP compliance testing is Ixia’s ANVL library (see https://www.ixiacom.com/products/ixanvl). The Quagga BGP stack that is integrated with the OPNFV’s BGPVPN feature is regularly tested using ANVL, latest results can be found at https://www.opensourcerouting.org/compliance-test-results/.

The main objective of the OPNFV BGPVPN feature is however not standard-compliant implementation of the BGP protocol, but integration of multiple components into a working system. We are not aware of a standard for testing the success/validity of such integration effort.

2. If a compliance test suite exists for the upstream component being tested, the upstream test suite should be used as a baseline. Please describe the baseline upstream suite. If this is not followed, or a subset is excluded or modified from the upstream, please provide justifications. An exception is required.

The BGPVPN tempest tests are not part of RefStack, so there is no upstream compliance testing for BGPVPN.

3. Test cases must be able to pass at least some OPNFV reference deployment or deployments, as deployed according to one or more scenarios in Danube. Please describe the scenarios that the test cases can pass (assuming the scenarios are correctly deployed - i.e. if it is due to errors of the SUT software, we don’t disqualify a test case for that).

The BGPVPN tests require the os-odl_l2-bgpvpn-(ha|noha) scenario to be deployed, which is currently supported by Fuel and Apex installers.
4. The features being tested must not rely on unmerged patches in the upstream projects. In other words, unmerged patches are considered not mature enough for Dovetail suite. Please list the upstream projects and affirm this is the case to the relevant upstream projects.

The BGPVPN feature does not require patches that are not yet merged upstream.

Notably, we consume a fork of the open source BGP stack Quagga, since the additions required to interface with the VPN Service implementation in ODL are in the merge backlog of the official Quagga repository for a long time. Recently, a part of the Quagga community has decided to fork Quagga and continue developing it in an independent project. We plan to transition to this new, more "official" fork during the Euphrates release.

The list of upstream components and their respective versions is:

- OpenStack with BGPVPN extension
- OpenDaylight Boron SR3 with new Netvirt feature enabled
- Quagga from 6Wind github
- OVS 2.6

5. The features being tested must be in scope of CVP during the Danube cycle. Please affirm that this is the case for the proposed set of test cases.

Unable to determine due to lack of information from CVP.

6. Any other important information or comments.

None.