Network Working Group
Internet-Draft
Intended status: Informational
Expires: 14 April 2022

Y.Mu
CAICT

14 April 2022

Guidelines for Network Operating System Testing
draft-mu-nos-testing-00

Status of This Memo

Copyright Notice

Abstract

This document presents a list of tests for implementers of Network Operating System(NOS) compliant Processes. This document specifies guidelines for a series of tests that can be run to probe the conformity

and robustness of the NOS implementations. These tests cover several important functions, in order to gain a level of confidence in the NOS implementation.

Table of Contents

1. Introduction 2
1.1. Document Scope 2
2. Conventions and Definitions 2
2.1. Conventions Used in This Document 2
2.2. Definitions 3
3. Test Specifications 3
3.1. DHCP Relay Agent Test 3
3.2. FIB Scale Test 4
3.3. IPv4 Decapsulation Test 5
3.4. LAG Feature Test 7
3.5. VLAN Feature Test 8
4. Security Considerations 9
Acknowledgments 9
References 9
Author's Address 10

1. Introduction

This memo describes a possible testing guideline for NOS implementations. The tests are intended to help demonstrate the conformity and robustness of the NOS implementations, and to illustrate common implementation errors. They are not intended to be an exhaustive set of tests and passing these tests does not necessarily imply conformance to the complete NOS specification.

Care should be taken regarding several test cases, as they might impact other systems in the network. We recommend that these specific tests should be executed only in a testbed environment.

The tests can be executed in a physical testbed environment or in a virtual testbed environment. A virtualized NOS testbed that can simulate a running NOS device and accompanying network topology using KVM and Docker is supported. This test setup does not require any physical switching hardware or any vendor SAI.

1.1. Document Scope

This document lists tests intended to be performed on an implementation of NOS. For some tests, multiple instances of each of those components are involved. The testing of those different NOS components complicates the testing as usually one tests his software against an existing implementation, which is proven to be compliant. The tests for NOS range from DHCP relay agent to FIB scale, IPv4 decapsulation, LAG feature, and VLAN feature. This document is not intended as a replacement for formal testing software procedures but as a best-practices approach to an informal testing of a developer's NOS implementation.

2. Conventions and Definitions

2.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Definitions

NOS -- Network Operating System
DHCP -- Dynamic Host Configuration Protocol
FIB -- Forward Information dataBase
IPv4 -- Internet Protocol version 4

LAG -- Link Aggregation Group
VLAN -- Link Aggregation Group

3. Test Specifications

The tests described in this section MAY be performed using a physical testbed environment or in a virtual testbed environment. The configuration of the DHCP, FIB, IPv4, LAG, and VLAN SHOULD be recorded for every test along with the test results.

The successful execution of all tests described in this section will give the tester a high confidence that the tested implementation is conformant with the NOS architecture and protocol. It does however not provide a 100% comprehensive coverage or formal proof of conformance.

3.1. DHCP Relay Agent Test

This section lists test aiming at confirming proper operation of certain features of the isc-dhcp-relay implementation. We must ensure the basic objective the DHCP relay agent is met. This involves validating that DHCP discover, offer, request, and ack packets are successfully relayed from client to server(s). We must also insure that features such as the proper attachment of an Option 82 field are supported by this agent.

The test simulates a DHCP client acquiring a lease. We simulate a DHCP client (a server under the ToR) booting up by manually creating DHCP packets and sending them on an interface that is part of the ToR's VLAN.

1. Create a DHCPDISCOVER packet and send it to the ToR on an interface that is part of the ToR's VLAN. The DHCP relay should receive this packet, add and Option 82 header and forward the packet to all known DHCP servers via it's uplinks.

2. Listen to all packets on the ToR's uplinks and count the number of forwarded DHCPDISCOVER packets to ensure one is sent to every known DHCP server.

3. Create a DHCPOFFER packet and send it to the ToR via one of its uplinks. The DHCP relay should forward this packet to the "client" on its VLAN.

4. Listen for the DHCPOFFER on the "client's" interface to ensure it would be received.

5. Create a DHCPREQUEST packet and send it to the ToR on an interface that is part of the ToR's VLAN. The DHCP relay should receive this packet, add and Option 82 header and forward the packet to all known DHCP servers via it's uplinks.

6. Listen to all packets on the ToR's uplinks and count the number of

forwarded DHCPREQUEST packets to ensure one is sent to every known DHCP server.

7. Create a DHCPACK packet and send it to the ToR via one of its uplinks. The DHCP relay should forward this packet to the "client" on its VLAN.

8. Listen for the DHCPACK on the "client's" interface to ensure it would be received.

DHCP servers should be defined in the device's minigraph file as a sub-object of each VLAN interface. The number of DHCP servers you define is up to you and your testing requirements.

3.2. FIB Scale Test

The purpose of this test is to test addition of IPV4 and IPV6 routes in a quantity required by NOS, and verify that each route is working properly by forwarding packets according to each route. The test assumes all routes are set prior by BGP, so no configuration is required to be done by the test itself. The test receives the route configuration via a text file to be parsed by the test. The validation to the routes is done by sending traffic on each of the created routes.

The Setup will be configured to have 6K IPV4/26 and 6K IPV6/64 routes. For each route a simple traffic class will be issued using the ping functionality.

For ECMP validation we'll use TCP packet, then for each ECMP route, we send n packets, then verify them to be received at any of ports in the ECMP group only.

As ECMP routes are to be set we should run few ping packets for each route.In additional each route can be verified independently thus test might take more than few minutes to cover all 12K route by few consecutive ping requests.

There will be 1 or 2 next hop groups per route.

The test is targeting a running NOS system with fully functioning configuration. The purpose of the test is not to test specific SAI API, but functional testing of routes, making sure that routes pre-configured on NOS start-up function properly.

The setup tests assumes to have single NOS connected to a switch connected to a server running 32 Arista VMs.

There will be 32 BGP peers connected to the switch. Each peer will have 2 BGP sessions open with the switch: single IPV4 connection and single IPV6 connection. The peers will advertise routes that switch needs to become aware of.

There will be 32 BGP peers connected to the switch. Each peer will have 2 BGP sessions open with the switch: single IPV4 connection and single IPV6 connection. The peers will advertise routes that switch needs to become aware of.

PTF host needs to be connected to a port through which it will send packets to the switch, and needs to have connection via ports through which the switch will send forward received packet back to the host for validation.

The peers and NOS will be deployed by an Ansible script. As part of deployment the script will generate the routes. For test preparation the same script will also generate a text file, route_info.txt, which will contain rows of following format: dst_prefix,port:port...

This information will be used to: 1. Create packet with proper destination prefix; 2. During validation expect packet from the list of ports.

The test objective is to validate that each route has been added to the switch and is functioning properly.

Test description:

1. Read information from route_info.txt for each route.

2. For each route in the route_info.textConstruct packet with proper destination address.
   - o Send 10 ping requests to the switch.
   - o Validate that 10 ping reply arrived on the port specified in route_info.txt for given destination address.
   - o If not all packets arrived, debug info extract from the switch will be printed to test log. Failure to be reported.

3. Test case summary to be printed in test log: number of routes to be sent, number of routes validated and found OK.

3.3. IPv4 Decapsulation Test

This test case is aimed at testing the ability to do de-capsulation of IP encapsulated packets, and verify that each decapsulated packet is with the right properties in each field, and forward with the corresponding underlay destination IP to the correct route. The test assumes all routes and decapsulation are set prior by to test, so no configuration is required to be done by the test itself, and the test will correspond only to the right IPs that is configured in the test.

The validation to the routes and the decapsulation is done by sending packets with the corresponding IPs both, in the overlay and underlay.

The scope of this test plan is only the Ansible test, including the PTF test and the necessary configuration.

The setup tests assume to have single NOS connected to a switch connected to a server running 32 Arista VMs.

There will be 32 BGP peers connected to the switch. The peers will advertise the default route and update the switch.

PTF host needs to be connected to a port through which it will send packets to the switch and needs to have a connection via ports through which the switch will send forward received packet back to the host for validation.

The peers and NOS will be deployed by an Ansible script. As part of the deployment, the script will generate the routes and decapsulation commands. The decapsulation rule will be generated by J2 script that will output JSON file with Decap IP that will configure through the SWSS config tool.

The test objective is to validate decapsulation ability and each route has been added to the switch and is functioning properly with the decapsulated packet.

Test configurations:

- IP decap IPv4 that will be taken from loopback IP: _Decap IP
- default IPv4 routes that will be configured through the BGP session as ECMP routes.
- unicase IPv4 routes that will configure throught the BGP session as TOR routes.

Test description:

1. The test will use host IP that fall into the default route and for the TOR routes.

2. The test will use different outer and inner TTL value combinations for different TTL modes.

3. From the PTF docker, craft and sent through all the ports a double encapsulated IP packets.

4. Verify the Sonic does not see the encapsulated packet. the IP-in-IP packet should not go to CPU, the packet should not be seen on the NOS.

5. Confirm that the packet that comes back to PTF Docker decapsulated from one of the expected ports.

6. repeat steps 1-4 32 times, so each port will send 2 packets one for unicast route and one ecmp route.

3.4. LAG Feature Test

This LAG feature test suite is targeting on testing basic LAG feature functionalities on NOS. This test suite includes several basic tests - FIB test, min-link test, and LACP test. Each test covers a basic functionality of LAG feature and ensures the switch works as expected under production scenarios.

The test is targeting a running NOS system with fully functioning configuration. The purpose of the test is not to test specific SAI API, but functional testing of LAG on NOS, making sure that traffic flows correctly, according to BGP routes advertised by BGP peers of NOS switch, and the LAG configuration. Test will be able to run only in the testbed specifically created for LAG.

Test configuration:

1. 8 LAGs per switch from the DUT to 8 EOS devices.

2. Each of the lag contains 2 members and the min-links is set to 2.

3. BGP sessions:
   - o 16 front panel ports north bound towards spine devices
   - o 16 front panel ports combine each two to have 8 LAGs south bound towards spines.

4. All TORs advertise 6 routes and all spine routers advertise 6402 routes. It is similar to the test environment set up for leaf devices without lags.

Test case -- Verify TCP traffic

Verify traffic between legs evenly distributed. Traffic is forwarded by NOS DUT.

- o PTF host will send packets according to the route_info.txt - will create packets with dst_ip according to route prefixes.
- o When packet reaches to SONIC DUT, it will route packet according to BGP routes, and send it to one of vEOS BGP peers.

- PTF test will receive a copy of the packet and perform validations described in Validation of Traffic

We are not targeting testing traffic coming into the DUT from BGP peers.

Test case -- LACP verification

The purpose of this test is to make sure that the LACP rate is correctly negotiated and set on both ends of the LAG.

Following are pre-conditions for the test case:

- The DUT switch is always started with the LACP rate set to 'slow'.
- The VMs are always set with rate 'fast' on startup.
- After startup all VMs will negotiate LACP rate with DUT and set their own rate to 'slow'.

The validation will be implemented as Ansible instructions in lag.yml, without invoking PTF:

1. Ansible playbook connects to each VM.

2. Ansible playbook validates that each VM has LACP rate set to 'slow'.

3. Ansible connects to DUT and validates LACP rate is set to 'slow'.

3.5. VLAN Feature Test

The purpose of VLAN feature test is to test VLAN functions on the NOS switch. This test suite includes several basic tests - FIB test, VLAN traffic test, link-flap test, and ARP learning test. Each test covers a basic functionality of VLAN feature and ensures the switch works as expected under production scenarios.

The tests will include:

1. Functionalities of VLAN ports.

2. VLAN interfaces routing.

3. IP2me traffic on VLAN interfaces.

The test will trying to cover all functionalities of VLAN ports including Ethernet ports and LAGs. And will make sure the IP traffic and IP2me traffic is working well.

A VLAN port will include three attributes:

- PVID: Ingress untagged packets will be tagged with PVID, and PVID will always in Permit VLAN IDs.
- Permit VLAN IDs: Which VLAN ID of ingress and egress packets is allowed in the port.
- tagged VLAN IDs: Determine which VLAN IDs egress packets will be tagged.

For the VLAN trunk feature, the tagged VLAN IDs are limited to Permit VLAN IDs besides PVID, e.g., if PVID is 100, Permit VLAN IDs are 100, 200, 300, tagged VLAN IDs are 200,300, in other words, untagged VLAN ID is 100.

Test description:

- The tests assume fanout switch support QinQ (stacked VLAN), so that stacked VLAN packets can passthrough fanout switch and can be tested on DUT with inner VLAN.
- Tests will be based on *t0* testbed type. The IP address of every LAGs on the DUT will be flushed to make all LAGs act as L2 ports. New test IP addresses will be configured on VLAN interfaces.
- VMs are only used to do LACP negotiation for LAGs; PTF is used to send packet and verify VLAN functionalities.

4. Security Considerations

This memo raises no security issues.

Acknowledgments

The authors wish to thank xxxx.

References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Authors' Addresses

Yubo Mu
China Academy of Information and Communications Technology
No.52, Hua Yuan Bei Road
Haidian District, Beijing
China

Phone: 010-62300556
EMail: muyubo@caict.ac.cn