Net Research

NetVision Compliance Tester (NVCT)

Freeware version

User's Guide

v.0.4 March 2017

User's Guide **1**
v.0.4 March 2017 1
Introduction **7**
System description **9**
Installation **12**
Configuration **13**
DUT identification 13
datapath_id 13
Basic DUT properties 13
aux_id 13
main_table_id 13
max_buffers 14
meter_max_n 14
miss_send_len 14
n_tables 14
unavailable_of_port 14
DUT description 15
adv_features 15
curr_features 15
curr_speed 15
dp_desc 15
hardware_desc 15
manufacturer_desc 16
max_speed 16
peer_features 16
serial_num 16
software_desc 16
supp_features 16
Advanced DUT properties 17
buffer_documentation_provided 17
buffer_size 17
buffer_timeout 17
connection_dead_interval 17
default_table_miss_behavior 17
fail_mode 17
pki_configured 18
ssl_timeout_interval 18
supported_experimenter_id 18
unsupported_instruction_id 18

unsupported_masked_match_id 18
unsupported_experimenter_id 19
unsupported_experimenter_type 19
DUT support for OpenFlow features 19
arbitrary_masks_supported 19
buffers_supported 19
bundles_supported 19
groups_supported 19
logical_interfaces_supported 20
match_prerequisites_supported 20
metadata_supported 20
meters_supported 20
queues_supported 20
roles_supported 20
set_field_supported 20
table_features_mod_enabled 21
table_features_mod_supported 21
table_features_config_enabled 21
table_features_config_supported 21
table_miss_config_supported 21
version_bitmaps_supported 21
Test properties 22
echo_connection_check 22
flow_add_max_threshold 22
flow_latency_test_repetitions 22
long_tests_allowed 22
manual_tests_allowed 23
meter_add_max_threshold 23
performance_tests_allowed 23
preferred_410_test_tunnel_type 24
sleep_time_1 24
sleep_time_2 24

Operations 24
Running the tests 25
Tester - Command Line Arguments 25
--remote-dataplane, -rd 25
--local-dataplane, -ld 26
--dpid, -d 26
--interface, -i 26
--no-port-detection, -npd 26
--remote-controller, -rc 26

--local-controller, -lc 26
--controller, -c 26
--local-address, -la 26
--openflow_cases, -of 27
--tests, -t 27
--no-sync, -ns 27
--quick, -q 28
System messages 28
manual tests disabled 28
expected value not configured 28
test dependency requirements not met 28
version bitmaps not supported 28
secure mode only test 28
standalone mode only test 28
test skipped due to estimated long execution time 28
match prerequisites not supported 29
no actions eligible for test detected 29
all tables supported 29
unsupported instruction not specified 29
only single table supported 29
all metadata write bits supported 29
arbitrary masks supported 29
unsupported masked match type not specified 29
unavailable openflow port id was not specified 29
group action not supported 30
set_field action not supported 30
meters action not supported 30
omitted due to the specification issues 30
default table-miss behavior config not available 30
immediate multipart replies detected 30
insufficient number of tables 30
performance tests disabled 30
optional counter not supported 30
buffers not supported 30
unsupported buffer timeout value 31
specified tunnel type not supported 31
experimenter parameters not specified 31
required field not found 31
could not generate unique mac address 31
expected error could not be triggered 31
table features config not supported 31
table features config not enabled 31

logical interfaces not supported 32
test for exclusive importance eviction mode 32
test for exclusive lifetime eviction mode 32
eviction not supported 32
eviction properties could not be modified 32
bundles not supported 32
meter mod failed 32
flow_mod failed 32
egress tables not supported 32
insufficient number of egress tables 33
schedule thresholds cannot be set 33
test result cannot be reliably verified 33
tests filtered by openflow version 33
System errors 33
dataplane connection failed 33
controller connection failed 33
no dataplanes discovered 33
no controller discovered 33
specified dpid not connected 33
no available dpid discovered 34
not enough interfaces specified 34
dataplane ntp sync error 34
controller ntp sync error 34
unsupported fail mode 34
System prompts 34
REBOOT DUT 34
CONFIGURE DUT TO CONNECT TO PORT port_number, SSL ENABLED/DISABLED 34
RESTORE PORT TO UP STATE… 34
DISCONNECT PORT port_number CABLE OR PUT THE PORT DOWN… 35
CONFIGURE DEFAULT TABLE-MISS BEHAVIOR TO PACKET-IN… 35
CONFIGURE DEFAULT TABLE-MISS BEHAVIOR TO DROP… 35
PROVIDE ID OF A CONFIGURED QUEUE OR SETUP QUEUE ON A SELECTED PORT AND ENTER ITS ID: 35
PROVIDE ID OF A PORT WITH CONFIGURED QUEUES OR SETUP QUEUES ON A SELECTED PORT AND ENTER ITS ID: 35
REMOVE QUEUES CONFIGURED DURING TEST… 35
SPECIFY EXPECTED CURR_FEATURES FOR PORT port_number: 35
SPECIFY EXPECTED ADV_FEATURES FOR PORT port_number: 36
SPECIFY EXPECTED SUPP_FEATURES FOR PORT port_number: 36
SPECIFY EXPECTED PEER_FEATURES FOR PORT port_number: 36
SPECIFY EXPECTED MAX_SPEED FOR PORT port_number: 36

SPECIFY EXPECTED CURR_SPEED FOR PORT port_number: 36
PROVIDE NUMBER OF QUEUES CONFIGURED ON PORT port_number: 37
PROVIDE TOTAL NUMBER OF QUEUES CONFIGURED ON A DEVICE: 37
PROVIDE ID OF A PORT WITHOUT ANY QUEUES CONFIGURED: 37
CONFIGURE FORWARDING TABLE: destination_mac AVAILABLE ON PORT DUT_port_id 37
CREATE SEPARATE VLAN vlan_id EXCLUDED FROM OPENFLOW INSTANCE… 37
REMOVE PREVIOUSLY CREATED VLAN… 37
MOVE INTERFACES interface_id_1 AND interface_id_2 TO VLAN vlan_id (TAGGED)... 37
RE-ADD PORTS TO OPENFLOW INSTANCE… 38
PROVIDE THE NAME OF OPENFLOW PORT port_id: 38
CONFIGURE tunnel_type TUNNEL (REMOTE_IP=ANY) AND PROVIDE TUNNEL KEY: 38
CONFIGURE tunnel_type TUNNEL (REMOTE_IP=UNIQUE PRIVATE SUBNET ADDRESS, KEY=ANY VALUE)... 38
PROVIDE TUNNEL LOGICAL PORT ID: 38
PROVIDE TUNNEL KEY: 38
PROVIDE TUNNEL REMOTE IP: 38
REMOVE PORT port_id (interface_name) FROM OPENFLOW INSTANCE… 38
CONFIGURE PORT port_id (interface_name) WITH UNIQUE IP ADDRESS OR ADD TO A MANAGEMENT VLAN… 39
CONFIGURE INTERFACE interface_name WITH UNIQUE IP ADDRESS FROM THE SAME SUBNET AS tunnel_remote_ip… 39
PROVIDE CONFIGURED PORT port_id (interface_name) IP ADDRESS (IP/MASK): 39
REMOVE ADDRESS FROM PREVIOUSLY CONFIGURED INTERFACE… 39
SWITCH DUT OPENFLOW VERSION TO {ver_id} (ver_name), RESET CONNECTION AND PRESS ENTER TO PROCEED… 39
RESET DATAPATH CONNECTION… 39
Example of system output 40

Specifications **43**
Tests Specification 43
Technical Specification 43
Licenses 44

Glossary **47**

1. Introduction

NetVision Compliance Tester (NVCT) is a complete solution prepared to support seamless implementation of the SDN enabled network components into the video aware IP networks. The core functionality of the NVCT is responsible for compliance testing of commercial of the shelf switches available at the market. NVCT enables comprehensive testing of the switches towards OpenFlow 1.3.4 specification and can provide clear comparisons of the products considered for implementation in the network. NVCT provides all key tests specified by ONF Conformance Test Specification for OpenFlow Switch Specification v1.3.4. in all 29 categories.

NetVision Compliance Tester is designed with regard to the future SDN networking functionality and evolving video distribution trends. In addition to 1.3.4 Test Specification functionalities, test for the most promising and useful SDN networking technologies are supported, including testing multiple forwarding tables pipeline, testing switches in hybrid mode and certification of VxLAN virtual networking.

Testing hybrid mode of operation has been identified as the key test procedure in the first stage of SDN implementation. SDN enabled switch typically handle incoming data streams using implemented and properly configured flow tables. However, it is possible to change its operational mode and use legacy L2 protocols designed for regular Ethernet switches. Both operational modes can be tested and verified with the NetVision Compliance Tester. Apparently NVCT offers VLAN-based test cases for L2/SDN hybrid switches.

Secure communication between controller and forwarding devices is the next key concern in regular data networks. NVCT provides carefully planned test suits of encrypted TLS connection between controller and DUT. Our tests can verify not only proper use of encryption protocols but also behavior of tested device under scenario assuming loss of secure connection to the controller.

For the most demanding SDN network components NVCT offers unique Performance Test Suite. Within the test suite NVCT provides functionality to discover number of forwarding tables installed in DUT as well as their capacity. This test is the performance indicator allowing comparison of different switch models. Additionally, NVCT enable delay measurements of selected flow modification actions. Future evolution of the SDN concept is in the centre of the NetVision Compliance Tester. NVCT provides tests for the key future SDN functionalities.

Detailed analysis of the OpenFlow 1.4 and 1.5 specifications resulted in novel test suites added to the current product version. NVCT tests switches under condition of forwarding table entries overflow, and verifies features like reporting low number of free tables entries (Eviction/Vacancy reports) as well as reasons of flow removal. Another important NVCT feature is related to testing message bundling. NVCT is also ready to test switch compliance with selected functionalities of the most recent OpenFlow 1.5 specification. NVCT can verify

implementation of egress tables in the tested device, this new flow handling scheme promises advanced functionality. However, early versions of egress table implementation are potential sources of serious errors in flows processing. NVCT offers tests designed towards testing egress tables functionality as well as ingress - egress flow handling. NVCT is also able to test extended flows statistics reports including right implementation of statistics triggers. This important extension of OpenFlow protocol enables intelligent network monitoring and was identified as the key functionality of future SDN network management systems. Finally, scheduled bundling and copy field actions test suits were implemented in current NVCT release.

NVCT can be installed and utilized in fully virtualized environments in order to ensure highest flexibility, the best user experience and cost effectiveness. NVCT is the key product for all the partners planning network evolution towards SDN architecture. NetResearch offers an excellent SDN testing product with reliable support of highly qualified network engineers. User should note that NVCT was designed to test OpenFlow devices in the laboratory environment while not under fully operational circumstances in production environment. Please note that in the freeware version only selected tests are available.

2. System description

The architecture overview is presented in the Fig. 1. The main components of the NVCT system are: in the dataplane NVCT traffic generator and analyzer, in the control plane SDN controller integrated with the NVCT brain. Device Under Test (DUT) is a switch pretending to be compliant with the OpenFlow protocol (OpenFlow Switch Specification).

Device Under Test (DUT) is stressed by the load originated in the NVCT traffic generator. After handling the traffic DUT sends it to the NVCT traffic analyzer in order to compare frames with expected ones. Dataplane logical connections between generator, DUT and analyzer may denote multiple physical connections, depending on the test configuration. It is not the case for control channel connections between DUT and OpenFlow controller, as well as connections between NVCT main component and SDN controller, NVCT generator and analyzer. In the control plane single physical connection is assumed. Between DUT and SDN controller OpenFlow protocol is utilized while SDN controller provides an API for the purpose of communication with NVCT main component. Finally, designed NVCT API is used for communication between traffic generator and analyzer modules with NVCT main component.

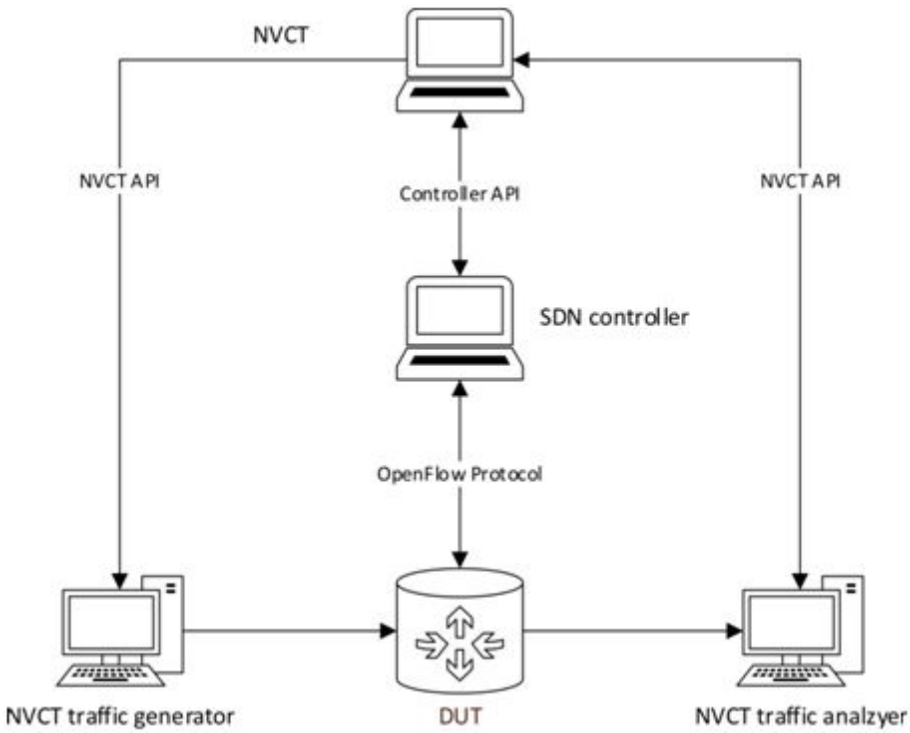

Fig. 1 Logical architecture of NVCT system

Fig. 2 presents system architecture including assignment of logical system components to the three virtual machines (VM#1, VM#2 and VM#3) composing NVCT system. VM#1 hosts the brain of NVCT system that integrates the whole process of testing. It is integrated with module responsible for analysis of control channel between SDN controller and DUT. Both the main component as well as analyzer are written in Python programming language. All

the components hosted by VM#1 may be further referred as *nvct-tester* or *tester*. VM#2 hosts open source SDN controller, the Ryu controller also developed in Python was modified for the system and integration purposes. The main responsibility of Ryu is to generate OpenFlow messages to the DUT. This customized SDN controller is further referred as *nvct-controller* or *controller*. Finally, VM#3 hosts both data plane components: traffic generator and analyzer. For the purpose of generation frames with designed headers Scapy Python library was used. Components hosted by the VM#3 are further referred as *nvct-dataplane*, while physical connections between VM#3 and DUT are referred simply as *dataplane*. The main advantage of distributing functionalities among virtual machines is scalability and simplified integration, deployment and updates of separated logical components.

All key tests from the 29 categories in ONF Conformance Test Specification for OpenFlow Switch Specification ver. 1.3.4 were implemented. Skipping single tests was caused mainly due to the documentation flaws. In addition to the ONF tests specification Net Research designed and implemented 129 testsin 7 categories (including tests designed for OpenFlow Protocol in versions 1.4 and 1.5). Detailed specification of those original tests was attached to this user guide in Specifications Section. Please note that in the freeware version only selected tests are available and full Specification is provided only for those tests. Tests available in the full version are only listed.

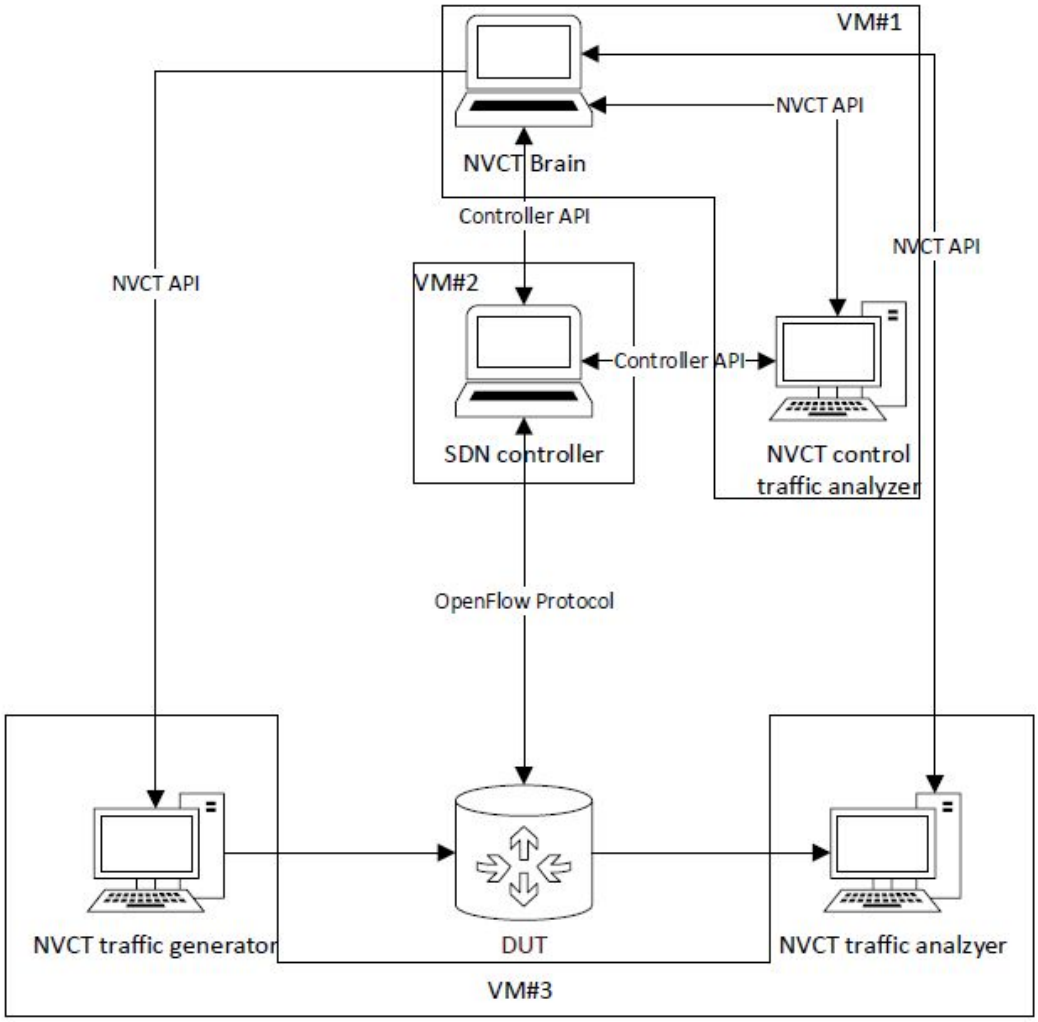

Fig. 2 Detailed architecture of NVCT system

We ensure easy and remote installation process of NVCT system. Each module can be separately installed using well known UNIX-based installation techniques. System was tested on selected hardware platforms, but on customer demand system may be easily prepared for other platforms. Installation Section describes the installation process carefully as well as supported hardware platforms. After installation some configuration process is needed. The most convenient option is to provide configuration details using dedicated configuration file. Otherwise, command line interface (CLI) will prompt for missing configuration. Therefore, the more configuration is provided before performing tests the more automatic is the test procedure. The whole process is described in the Configuration Section. CLI is also used to run tests and for the operational purposes. In some cases CLI may prompt user for an action needed to perform the test (e.g. unplug the cable). All possible actions and other operational details are described in Operations Section. Thus, any network engineer and administrator will be able to install and configure NVCT system, as well as run and operate tests using it.

3. Installation

Install NVCT modules (*nvct-tester, nvct-controller, nvct-dataplane*) from provided packages (.deb or .rpm), e.g.:

```
nn@nvct:~$ sudo dpkg -i nvct-controller_1.0-1_amd64.deb
nn@nvct:~$ sudo rpm -i nvct-controller_1.0-1.amd64.rpm
```

Run *nvct-controller* and *nvct-dataplane* first, then launch *nvct-tester* to perform conformance test.

```
nn@nvct:~$ nvct-controller
nn@nvct:~$ sudo nvct-dataplane
nn@nvct:~$ nvct-tester
```

All config files and output files, including results report, will be created in current working directory. Results report will be also sent to user's e-mail address if configured. Supply proper command line arguments to configure additional features.

```
nn@nvct:~$ nvct-tester -t onf_013_010_control_channel -of 13
--dpid 64
```

Place and specify certificates and private keys for *nvct-controller* to perform SSL connection tests properly.

CLI arguments:

--cert-dir - certificate and keys directory (default ".")
--ctl-privkey - controller's private key filename (default "controller-privkey.pem")
--ctl-cert - controller's certificate filename (default "controller-cert.pem")
--ca-cert - switch CA certificate filename (default "switch-cacert.pem")
-of <openflow version> - limit available OpenFlow modes to single version (for incompatible switches, e.g. -of 13) (disabled by default)

```
nn@nvct:~$ nvct-controller --cert-dir ~/certs/
```

```
nn@nvct:~$ nvct-controller --ctl-cert ctlcert.pem --ca-cert
cacert.pem --ctl-privkey ctlkey.pem
```

For instructions on retrieving proper certificates from PKI refer to your device's User Manual.

It is possible to run NVCT under any virtualization platform, e.g. VMware, KVM or VirtualBox

4. Configuration

Configuration parameters introduced in this section should be defined in the *dut_config.cfg* file placed in the main directory of the system. Notation is as follows: *parameter_name*: *parameter_value*. Parameters are logically classified into six groups, in each group parameters are described in alphabetical order. For each parameter valid values are provided, as well as default value and detailed description. Additionally, particular tests dependent from the configuration parameter are listed when applicable. Parameter is assigned *None* value when it was not configured.

The configuration file is automatically created in current working directory. Unless, -q parameter is specified, user is prompted for missing configuration values before test procedures start. All values should be input with regards to constraints specified in the documentation below. Several values are required for specific test cases. If not provided by the user, dependant test cases are skipped due to insufficient configuration data. In case when all configuration parameters are provided in configuration file, user will be prompted only in case of tests that requires real time interaction with user (e.g. please unplug the cable, or configure feature within the DUT).

4.1. DUT identification

4.1.1. datapath_id

values: numerical value
default: *None*
Field uniquely identifies a *datapath*, it is OpenFlow Datapath ID which identifies each OpenFlow instance. The lower 48 bits are intended for the switch MAC address, while the top 16 bits are up to the implementer and should be manually configured by the user

4.2. Basic DUT properties

4.2.1. aux_id

values: numerical value
default: 0
Identifies the type of connection from the switch to the *controller*. Each connection from the switch to the *controller* is identified by the switch Datapath ID and an Auxiliary ID. The main connection has this field set to zero, an auxiliary connection has this field set to a non-zero value, it must be configured to perform test: onf_013_040_040_auxiliary_id

4.2.2. main_table_id

values: numerical value
default: 0

Id of main table used for majority of tests

4.2.3. max_buffers

values: numerical value
default: *None*
Number of packets that DUT can buffer at once according to the DUT specification provided by the vendor, it must be configured to perform test: onf_013_040_020_max_buffers

4.2.4. meter_max_n

values: numerical value
default: *None*
Maximum meter capacity supported by the DUT, it must be set in order not to skip the onf_013_440_580_meter_mod_failed_out_of_meters, onf_013_440_600_meter_mod_failed_data tests in case when error was not triggered

4.2.5. miss_send_len

values: numerical value
default: 128
Field defines the number of bytes of each packet sent to the controller by the OpenFlow pipeline when not using an output action to the OFPP_CONTROLLER logical port, for example sending packets with invalid TTL if this message reason is enabled. OpenFlow Protocol specification defines default value to equal 128. It must be configured to perform test: onf_013_040_120_miss_send_len

4.2.6. n_tables

values: numerical value
default: *None*
Number of tables that DUT support by datapath according to the DUT specification provided by the vendor, it must be configured to perform tests:
onf_013_040_030_number_of_tables, onf_013_320_010_table_statistics_count, onf_013_320_020_identifier_of_table, onf_013_320_030_active_entries, onf_013_320_040_packets_looked_up, onf_013_320_050_packets_hit, onf_013_320_140_unique_id, onf_013_320_190_table_order, nvct_013_040_010_number_of_tables. In order to perform tests: nvct_013_050_010_pipeline_table_0, nvct_013_050_020_pipeline_table_gt_0, nvct_013_050_030_table_gt_0_default_action, nvct_013_050_040_skip_tables, nvct_013_050_050_goto_table_lt, nvct_013_050_060_goto_table_last_table, nvct_013_050_070_table_miss_chain,

4.2.7. unavailable_of_port

values: numerical identifier
default: *None*
User configure this parameter to point out the id of OpenFlow port that is unavailable, it must be set to perform test: onf_013_150_160_currently_invalid_port. User should refer to the DUT manual in order to make OpenFlow port unavailable

4.3. DUT description

4.3.1. adv_features

values: numerical identifier
default: *None*
Defines expected features being advertised by the port in a description of particular standard port of the DUT. If value not configured and *manual_tests_allowed* configuration parameter is *True* user will be asked to provide the value of the parameter. Otherwise, following tests will be skipped: onf_013_340_280_advertised_features. User should refer to the DUT manual in order to provide proper value.

4.3.2. curr_features

values: numerical identifier
default: *None*
Defines expected current features in a description of particular standard port of the DUT. If value not configured and *manual_tests_allowed* configuration parameter is *True* user will be asked to provide the value of the parameter. Otherwise, following tests will be skipped: onf_013_340_270_current_features. User should refer to the DUT manual in order to provide proper value.

4.3.3. curr_speed

values: numerical value
default: *None*
Defines expected current port bitrate in kbps according to the description of particular standard port of the DUT. If value not configured and *manual_tests_allowed* configuration parameter is *True* user will be asked to provide the value of the parameter. Otherwise, following tests will be skipped: onf_013_340_310_current_bit_rate. User should refer to the DUT manual in order to provide proper value.

4.3.4. dp_desc

values: string of max 256 ASCII characters
default: *None*
Free-form string to describe the datapath for debugging purposes, e.g., “switch3 in room 3120”, expected value should be provided by the vendor, it must be configured to perform test: onf_013_040_210_dp_description

4.3.5. hardware_desc

values: string of max 256 ASCII characters
default: *None*
Information about the underlying hardware, expected value should be provided by the vendor, it must be configured to perform test: onf_013_040_180_hardware_description

4.3.6. manufacturer_desc

values: string of max 256 ASCII characters
default: *None*
Information about the DUT manufacturer, expected value should be provided by the vendor, it must be configured to perform test: onf_013_040_170_manufacturer_description

4.3.7. max_speed

values: numerical value
default: *None*
Defines expected maximum port bitrate in kbps according to the description of particular standard port of the DUT. If value not configured and *manual_tests_allowed* configuration parameter is *True* user will be asked to provide the value of the parameter. Otherwise, following tests will be skipped: onf_013_340_320_max_bit_rate. User should refer to the DUT manual in order to provide proper value.

4.3.8. peer_features

values: numerical identifier
default: *None*
Defines expected features being advertised by the peer in a description of particular standard port of the DUT. If value not configured and *manual_tests_allowed* configuration parameter is *True* user will be asked to provide the value of the parameter. Otherwise, following tests will be skipped: onf_013_340_300_peers_features. User should refer to the DUT manual in order to provide proper value

4.3.9. serial_num

values: string of max 32 ASCII characters
default: *None*
Information about the DUT serial number, expected value should be provided by the vendor, it must be configured to perform test: onf_013_040_200_serial_number

4.3.10. software_desc

values: string of max 256 ASCII characters
default: *None*
Information about the software revision, expected value should be provided by the vendor, it must be configured to perform test: onf_013_040_190_software_description

4.3.11. supp_features

values: numerical identifier
default: *None*
Defines expected features supported by the port in a description of particular standard port of the DUT. If value not configured and *manual_tests_allowed* configuration parameter is

True user will be asked to provide the value of the parameter. Otherwise, following tests will be skipped: onf_013_340_290_supported_features. User should refer to the DUT manual in order to provide proper value

4.4. Advanced DUT properties

4.4.1. buffer_documentation_provided

values: *True/False*
default: *False*
Setting this parameter to True user confirms that documentation for the DUT buffer configuration was provided with the DUT by the vendor

4.4.2. buffer_size

values: numerical value in bytes
default: 256
Defines the size of a buffer in bytes

4.4.3. buffer_timeout

values: numerical value in seconds
default: 5
Defines how long packets can be stored in the DUT internal buffer. If value is too small test onf_013_410_090_packet_in_buffer_timeout will be skipped

4.4.4. connection_dead_interval

values: numerical value in seconds
default: 10
Time required for connection to be detected as inactive, used in channel failure simulations.

4.4.5. default_table_miss_behavior

values: string with expected values: drop or packet-in
default: *None*
Indicates what is the default table-miss behaviour of the DUT. One of the expected values must be configured to perform test: nvct_013_050_030_table_gt_0_default_action, User should refer to the DUT manual to configure default table miss behaviour

4.4.6. fail_mode

values: *secure/standalone*
default: *None*
It must be configured to perform tests:
onf_013_010_010_startup_behavior, onf_013_010_100_failure_mode, onf_013_010_110_fail_secure_mode, onf_013_010_120_fail_standalone_mode, onf_013_010_130_entries_stay_active, and nvct_013_030_080_tls_fail_standalone_mode_behavior

It indicates DUT behaviour when network controller is not reachable. *Secure* value means that switch will continue to handle dataplane traffic according to the existing forwarding tables but will also drop all the packets that destined to the controller. In *standalone* mode switch starts to operate as a classic L2 switch. Tests onf_013_010_110_fail_secure_mode and onf_013_010_130_entries_stay_active will be performed if *fail_mode* parameter was set to *secure*. Tests onf_013_010_120_fail_standalone_mode and nvct_013_030_080_tls_fail_standalone_mode_behavior will be performed if *fail_mode* parameter was set to *standalone*. Assigning any other value may rise *unsupported fail mode error*.

4.4.7. pki_configured

values: *True/False*
default: *None*
Indicates if Public Key Infrastructure was properly configured in the DUT. It must be configured to perform test: onf_013_010_020_certificate_configuration. For details about PKI configuration in DUT please refer to the DUT manual. PKI must be configured and *pki_configured* parameter must be set to *true* in order to perform tests:
onf_013_010_050_default_tls_port, onf_013_010_060_non_default_tls_port

4.4.8. ssl_timeout_interval

values: numerical value in seconds
default: 10
Defines time after which DUT treats ssl session between DUT and controller as disconnected in case when controller is not responding

4.4.9. supported_experimenter_id

values: numerical identifier
default: *None*
Defines id of experimenter that is supported by the DUT, it must be set to perform test: onf_013_430_130_bad_experimenter_type, onf_013_440_680_experimenter_error_message

4.4.10. unsupported_instruction_id

values: numerical identifier
default: *None*
User configure this parameter to point out the OpenFlow instruction that is not supported by the DUT (if such an instruction exist), it must be set to perform test:
onf_013_150_050_unsupported_instructions

4.4.11. unsupported_masked_match_id

values: numerical identifier
default: *None*
User configure this parameter to point out the OpenFlow masked match type that is not supported by the DUT (if such a exist), it must be set to perform test
onf_013_150_130_unsupported_mask

4.4.12. unsupported_experimenter_id

values: numerical identifier
default: *None*
Defines id of experimenter that is not supported by the DUT, it must be set to perform test: onf_013_430_120_bad_experimenter

4.4.13. unsupported_experimenter_type

values: numerical identifier
default: *None*
Defines type of experimenter that is not supported by the DUT, it must be set to perform test: onf_013_430_130_bad_experimenter_type and onf_013_440_680_experimenter_error_message,

4.5. DUT support for OpenFlow features

4.5.1. arbitrary_masks_supported

values: *True/False*
default: *False*
Indicates if switch supports arbitrary bitmasks on specific match fields. These masks can more precisely specify matches, must be set to *False* to perform tests: onf_013_150_110_bad_error_mask, onf_013_150_120_nd_dl_mask_wrong and onf_013_150_130_unsupported_mask, and onf_013_430_690_bad_wildcard_match,

4.5.2. buffers_supported

values: *True/False*
default: *False*
Informs if DUT supports buffers that allows storing packets by the DUT when sending packets to the controller using packet-in messages, user should refer to DUT manual in order set proper value, it must be set to True to perform test: onf_013_410_090_packet_in_buffer_timeout

4.5.3. bundles_supported

values: *True/False*
default: *False*
Informs if DUT supports bundles, user should refer to DUT manual in order set proper value, must be set to *True* in order to performs tests: nvct_014_490_410_bundle_correct_usage, nvct_014_490_420_bundle_immediate_error, nvct_014_490_430_bundle_error_after_commit, nvct_014_490_440_ordered_messages, and nvct_014_490_450_bundle_echo_messages

4.5.4. groups_supported

values: *True/False*
default: *False*

Informs if DUT supports groups action which allows to direct flow to the group, user should refer to DUT manual in order set proper value, it must be set to *True* to perform test: onf_013_150_170_undefined_group

4.5.5. logical_interfaces_supported

values: *True/False*
default: *False*
Informs if DUT supports logical interfaces, user should refer to DUT manual in order set proper value, it must be set to *True* to perform tests: nvct_013_020_010_tunnel_id_gre, nvct_013_020_020_tunnel_id_mpls, nvct_013_020_030_tunnel_id_vxlan, nvct_013_020_100_vxlan_forwarding, and nvct_013_020_110_vxlan_set_field

4.5.6. match_prerequisites_supported

values: *True/False*
default: *False*
Indicates if DUT supports matching on prerequisite fields. This feature must be supported in order to perform tests: onf_013_080_180_missing_prerequisite , onf_013_080_190_prerequisite_on_wrong_position and onf_013_080_200_multiple_instances_of_the_same_oxm_type

4.5.7. metadata_supported

values: *True/False*
default: *False*
Informs if DUT supports metadata (register value that is used to carry information from one table to the next), user should refer to DUT manual in order set proper value

4.5.8. meters_supported

values: *True/False*
default: *False*
Informs if DUT supports meters that can measure and control the rate of packets, user should refer to DUT manual in order set proper value

4.5.9. queues_supported

values: *True/False*
default: *False*
Informs if DUT supports queues on OpenFlow ports, user should refer to DUT manual in order set proper value

4.5.10. roles_supported

values: *True/False*
default: *False*
Informs if DUT supports roles, user should refer to DUT manual in order set proper value

4.5.11. set_field_supported

values: *True/False*
default: *False*
Informs if DUT supports set field action which permits the rewriting of any header field in a single action, user should refer to DUT manual in order set proper value, it must be set to *True* to perform test: onf_013_150_180_bad_set_argument

4.5.12. table_features_mod_enabled

values: *True/False*
default: *False*
Informs if tables features modification is enabled on DUT, user should refer to DUT manual in order to disable/enable table features modification and to set proper value

4.5.13. table_features_mod_supported

values: *True/False*
default: *False*
Informs if DUT supports modification of tables, user should refer to DUT manual in order set proper value

4.5.14. table_features_config_enabled

values: *True/False*
default: *False*
Informs if tables features configuration is enabled on DUT, user should refer to DUT manual in order to disable/enable table features configuration and to set proper value

4.5.15. table_features_config_supported

values: *True/False*
default: *False*
Informs if DUT supports configuration of table features, user should refer to DUT manual in order set proper value, it must be set to *True* to perform tests: onf_013_440_610_table_features_bad_table, onf_013_440_620_table_features_bad_metadata, onf_013_440_630_table_features_bad_type, onf_013_440_640_table_features_bad_length, onf_013_440_650_table_features_failed_bad_argument, onf_013_440_670_table_features_failed_data

4.5.16. table_miss_config_supported

values: *True/False*
default: *False*
Informs if DUT supports default DUT behaviour in case of table-miss (packet is not matching any flows in the table), user should refer to DUT manual in order set proper value, it must be set to *True* to perform tests: onf_013_250_070_max_bytes_of_packet, and onf_013_250_140_miss_send_len

4.5.17. version_bitmaps_supported

values: *True/False*
default: *False*
Indicates if DUT supports bitmasks for the purpose of OpenFlow Protocol version negotiation with network controller. This feature must be supported in order to perform test: onf_013_010_090_version_negotiation_bitmap. In order to check if DUT supports the bitmaps please refer to the DUT manual.

4.6. Test properties

4.6.1. echo_connection_check

values: *True/False*
default: *False*
Indicates if *controller* will check connection availability with *dataplane*.

4.6.2. flow_add_max_threshold

values: numerical value
default: 100000
This value is used while stressing flow tables capacity, particular test may be skipped if number of flows added to the table exceeds this threshold and long tests are not allowed (*long_tests_allowed* configuration parameter set to *False*), simultaneously

4.6.3. flow_latency_test_repetitions

values: numerical value
default: 20
Defines number of repetitions while examining latency of some operations performed by the DUT, the more repetitions the more valuable is computed mean latency, but on the other hand the longer is time execution time

4.6.4. long_tests_allowed

values: *True/False*
default: *False*
Indicates if tests that are expected to be executed for a long time should be allowed. For comprehensive tests this parameter should be set to *True*. However, for fast preliminary testing it may be set to *False*, must be set to *True* to perform the test:
onf_013_180_410_duration_precision, onf_013_300_280_multipart_buffer_overflow,
nvct_013_040_020_capacity_all_header_fields, nvct_013_040_030_capacity_l2_header_fields,

nvct_013_040_040_capacity_l3_header_fields, nvct_014_490_110_eviction_disabled_table_full_error, nvct_014_490_120_eviction_of_entries, nvct_014_490_130_eviction_flags_importance, nvct_014_490_140_eviction_flags_lifetime, nvct_014_490_150_eviction_flags_importance_mod

4.6.5. manual_tests_allowed

values: *True/False*
default: *True*
Indicates if tests that need interaction with user will be performed. To test DUT carefully this option should be set to *True*. However, in case when user is not able to interact with system during tests it should be changed to *False*, it must be set to *True* to perform tests: onf_013_180_230_correct_packet_drop_counters, onf_013_200_070_port_status, onf_013_250_070_max_bytes_of_packet, onf_013_250_140_miss_send_len, onf_013_320_500_write_actions_miss, onf_013_320_510_apply_actions, onf_013_320_520_apply_actions_miss, onf_013_320_540_write_actions, onf_013_340_260_port_state, onf_013_380_040_queue_stats_reserved, onf_013_380_050_queue_stats_standard, onf_013_380_080_queue_stats, onf_013_380_090_queue_config_request_reserved, onf_013_380_100_queue_config_request_standard, onf_013_380_130_queue_configuration, onf_013_410_260_physical_port_match, onf_013_410_280_tunnel_id_match, onf_013_410_320_logical_port_match, onf_013_420_100_port_status_reason_add, onf_013_420_110_port_status_reason_delete, onf_013_420_120_port_status_reason_modify, nvct_013_010_010_normal_forwarding_predefined_rule, nvct_013_010_110_vlan_isolation_forwarding_predefined_rule, nvct_013_010_120_vlan_isolation_forwarding_mac_learning, nvct_013_010_130_vlan_isolation_forwarding_unicast_flood, nvct_013_010_140_vlan_isolation_forwarding_both, nvct_013_020_010_tunnel_id_gre, nvct_013_020_020_tunnel_id_mpls, nvct_013_020_030_tunnel_id_vxlan, nvct_013_020_100_vxlan_forwarding, nvct_013_020_110_vxlan_set_field, nvct_013_030_010_tls_default_tcp_port_encrypted_controller_to_switch_message, nvct_013_030_020_tls_default_tcp_port_encrypted_asynchronous_message, nvct_013_030_030_tls_default_tcp_port_encrypted_symmetric_message, nvct_013_030_040_tls_non_default_tcp_port_encrypted_controller_to_switch_message, nvct_013_030_050_tls_non_default_tcp_port_encrypted_asynchronous_message, nvct_013_030_060_tls_non_default_tcp_port_encrypted_symmetric_message, nvct_013_030_070_tls_session_timeout, nvct_013_030_080_tls_fail_standalone_mode_behavior, nvct_013_040_310_table_miss_performance_default_behavior,

4.6.6. meter_add_max_threshold

values: numerical value
default: 100
This value is used while testing max meter capacity of the DUT. Higher value should be assigned to this parameter than to the *meter_max_n* parameter in order not to skip the onf_013_440_580_meter_mod_failed_out_of_meters, onf_013_440_600_meter_mod_failed_data tests in case when error was not triggered

4.6.7. performance_tests_allowed

values: *True/False*
default: *True*
Indicates if performance tests will be performed. To test DUT carefully this option should be set to *True*, it must be set to *True* to perform tests:onf_013_320_240_max_entries, nvct_013_040_020_capacity_all_header_fields, nvct_013_040_030_capacity_l2_header_fields, nvct_013_040_040_capacity_l3_header_fields,nvct_013_040_110_single_flow_add_latency, nvct_013_040_115_single_flow_add_latency_occupied_table, nvct_013_040_120_flow_add_rate, nvct_013_040_130_wildcard_flow_add_rate,nvct_013_040_140_single_flow_modification_latency, nvct_013_040_145_single_flow_modification_latency_occupied_table, nvct_013_040_150_flow_modification_rate, nvct_013_040_160_single_flow_delete_latency, nvct_013_040_165_single_flow_delete_latency_occupied_table, nvct_013_040_170_flow_delete_rate, nvct_013_040_210_packet_out_handling, nvct_013_040_220_packet_out_size,

4.6.8. preferred_410_test_tunnel_type

values: string identifying tunnel type
default: ‘GRE’
Defines the preferred tunnel type used for the purpose of tests from packet in message test suite. If preferred tunnel type is not supported than following tests will be skipped: onf_013_410_260_physical_port_match, onf_013_410_280_tunnel_id_match. onf_013_410_320_logical_port_match

4.6.9. sleep_time_1

values: numerical value
default: 0.25
Defines sleep time common for all the tests in which system must wait for the DUT to perform some actions or for informations to propagate. In case of slow operation or long propagation time user may increase the sleep time. However, the higher value is assigned the longer is test execution time

4.6.10. sleep_time_2

values: numerical value
default: 0.25
Defines basic capture time common for all the performance tests. The longer value is set the more statistical valuable results are, but also, the longer is test execution time

4.6.11. report_recipient_mail

values: e-mail address
default: *None*
Defines recipient e-mail address for the report PDF file. If not specified, report will not be sent.

5. Operations

This section explains how to start all system components and how to run tests, also command line arguments to be provided when executing tests are carefully described. Additionally, all possible system messages are described and carefully explained. System messages explain behaviour of the system, they mostly refer to the reason explaining why particular test was skipped. System errors indicate that runtime error occurred, and the issue should be investigated in order to obtain valuable test results. Finally, system prompts ask user to take some actions in order to perform the test.

From the whole collection selected tests are marked as manual. Manual tests need some action from the user to be performed. Manual tests will be run as first group of tests. Thanks to that, user may set up whole test, interact with the system at the very beginning of testing process and after that no further action is needed till results are provided. Some tests are manual by design as interaction is needed during test execution. Selected tests may be

manual or automatic depending on configuration provided in *dut_config.cfg* file as explained in Configuration Section. Additionally, selected tests may be also marked as long tests in terms of execution time and as performance tests.

All test groups, manual, long and performance, may be separately enabled and disabled using *manual_tests_allowed*, *long_tests_allowed* and *performance_tests_allowed*, respectively. In order to test all the features of DUT user sholud configure all mentioned configuration parameters to *true*. However, when user is not able to interact with system after starting the test manual tests should be disabled. Analogously, long tests should be disabled in order to decrease tests execution time, while performance tests should be disabled in case of shortage in processing or network resources within the machine hosting NVCT system.

5.1. Running the tests

System description Section presents the architecture of the NVCT system. Three modules are introduced: *nvct-dataplane*, *nvct-controller* and *nvct-tester*. All three modules are installed in the system by the package manager as described in the initial section of the document. Each module should be launched separately, using respective commands: *nvct-dataplane*, *nvct-controller* and *nvct-tester*. Please note that *nvct-dataplane* and *controller* modules must be started before *tester* module. That is because *tester* module, introduced as a brain of the system, is responsible for actual tests execution and needs *dataplane* and *controller* to do so.

As *tester* module have to communicate with *nvct-dataplane* and *controller* modules it must be familiar with their IP addresses. To facilitate system configuration IP addresses autodetection feature has been implemented. Thus, by default, user do not have to configure any IP addresses in the context of the system. *Tester* module will be able to discover addresses of *nvct-dataplane* and *controller* modules and ensure communication between all the components. However, for the purpose of manual configuration of IP addresses corresponding command line arguments are available.

By default, *nvct-dataplane* module is remote for the *tester* module and such a case is described in System Description Section. However, it is possible to run both modules as a single application. For configuration details please refer to the Command Line Arguments Sction. Please note *nvct-dataplane* module must be run with root privileges.

5.2. Tester - Command Line Arguments

Command line arguments should be provided by the user when starting tests. Each argument has full name and shorter version. Both are provided and separated using comma. Default value is provided when applicable. Thus, schema to run the *tester* module is as follows:
path/nvct-tester --argument value -argument2_short_ver value

5.2.1. --remote-dataplane, -rd

In this argument IP address of the remote *nvct-dataplane* should be provided.

5.2.2. --local-dataplane, -ld

Indicates if local *nvct-dataplane* should be used. If true *nvct-dataplane* and *tester* modules are run as single application. Default: *true*.

5.2.3. --dpid, -d

OpenFlow datapath identifier of DUT. Each OpenFlow instance on a switch is identified by a Datapath Identifier. This is a 64 bit number determined as follows according to the OpenFlow specification. The lower 48 bits are intended for the switch MAC address, while the top 16 bits are up to the implementer. An example use of the top 16 bits would be a VLAN ID to distinguish multiple virtual switch instances on a single physical switch

5.2.4. --interface, -i

This argument is to bind interface name with OpenFlow interface id. Expected format is *name:of_id.*

5.2.5. --no-port-detection, -npd

To enable or disable automatic port detection. When enabled system finds IP addresses and ports of all system components and user do not have to configure those addresses manually.

5.2.6. --remote-controller, -rc

In this argument IP address of the remote *controller* module should be provided.

5.2.7. --local-controller, -lc

If *controller* module is run on localhost user may assign true to this parameter and system will automatically look for the *controller* on the 127.0.0.1 address. Default: *true*.

5.2.8. --controller, -c

In this argument IP address of the SDN *controller* should be provided. Default: 127.0.0.1

5.2.9. --local-address, -la

In this argument IP address of the *tester* module should be provided in case of invalid remote module detection.

5.2.10. --openflow_cases, -of

In this argument OpenFlow versions to be tested are provided. Expected values are: 13, 14 and 15. Those values denote OpenFlow 1.3, 1.4 and 1.5, respectively. To test numerous versions of OpenFlow all values should be provided, e.g. *-of 13 -of 14* (to test OpenFlow 1.3 and 1.4)

5.2.11. --tests, -t

Test cases to be performed. Expected values refer to the test cases (see Example of system output Section) and are as follows:

nvct.of_tests.basic,
nvct.of_tests.nvct_013_010_hybrid_mode,
nvct.of_tests.nvct_013_020_tunneling,
nvct.of_tests.nvct_013_030_security_features,
nvct.of_tests.nvct_013_040_performance,
nvct.of_tests.nvct_013_050_multiple_table_features,
nvct.of_tests.nvct_014_490_of14,
nvct.of_tests.nvct_015_490_of15,
nvct.of_tests.onf_013_010_control_channel,
nvct.of_tests.onf_013_040_controller_to_switch_messages,
nvct.of_tests.onf_013_050_flow_table_miss,
nvct.of_tests.onf_013_060_flow_table_matching,
nvct.of_tests.onf_013_080_flow_table_match_prerequisites,
nvct.of_tests.onf_013_090_flow_table_match_combinations,
nvct.of_tests.onf_013_100_flow_table_actions,
nvct.of_tests.onf_013_130_flow_table_action_set,
nvct.of_tests.onf_013_140_flow_table_modifications,
nvct.of_tests.onf_013_150_flow_table_errors,
nvct.of_tests.onf_013_180_counters,
nvct.of_tests.onf_013_200_protocol_messages,
nvct.of_tests.onf_013_210_port_structure_protocol_message,
nvct.of_tests.onf_013_230_action_header_protocol_message,
nvct.of_tests.onf_013_250_switch_config_protocol_message,
nvct.of_tests.onf_013_260_flow_mod_protocol_messages,
nvct.of_tests.onf_013_300_multipart_reply_protocol_message,
nvct.of_tests.onf_013_310_multipart_reply_one,
nvct.of_tests.onf_013_320_multipart_reply_two,
nvct.of_tests.onf_013_330_wildcards,
nvct.of_tests.onf_013_340_multipart_reply_four,
nvct.of_tests.onf_013_380_multipart_reply_five,
nvct.of_tests.onf_013_390_packet_out_protocol_message,
nvct.of_tests.onf_013_410_packet_in_message,
nvct.of_tests.onf_013_420_flow_removed,
nvct.of_tests.onf_013_430_error_messages_one,
nvct.of_tests.onf_013_440_error_message_two,
nvct.of_tests.onf_013_450_symmetric_messages,

5.2.12. --no-sync, -ns

If set to true disables clock synchronization. Default: false.

5.2.13. --quick, -q

Skip all configuration parameter prompts (including user notes) during setup phase. Default: false.

5.3. System messages

5.3.1. manual tests disabled

Informs user that particular test needs manual interaction with user and was skipped because *manual_tests_allowed* configuration parameter was set to *False*.

5.3.2. expected value not configured

Informs user that particular test was skipped due to the missing configuration of selected configuration parameter.

5.3.3. test dependency requirements not met

Informs user that particular test cannot be run as it is dependent from one of the previous tests which failed. In some cases natural dependencies between tests occur as some dependent functionalities are not reasonable or cannot be supported if other features are not compliant with OpenFlow Protocol.

5.3.4. version bitmaps not supported

Informs user that particular test was skipped as *version_bitmaps_supported* configuration parameter was set to *False*.

5.3.5. secure mode only test

Informs user that particular test was skipped as *fail_mode* configuration parameter was not set to *secure* while this test regards to testing fail-secure mode only.

5.3.6. standalone mode only test

Informs user that particular test was skipped as *fail_mode* configuration parameter was not set to *standalone* while this test regards to testing fail-standalone mode only.

5.3.7. test skipped due to estimated long execution time

Informs user that particular test is expected to execute for a long time and was skipped because *long_tests_allowed* configuration parameter was set to *False*.

5.3.8. match prerequisites not supported

Informs user that particular test was skipped as *match_prerequisites_supported* configuration parameter was set to *False*.

5.3.9. no actions eligible for test detected

Informs user that test was skipped as table under test do not support any of the actions while at least one action must be supported to perform the test.

5.3.10. all tables supported

Informs user that test was skipped as DUT reports that all the tables with identifiers within 0 - 254 range are supported. As table identifiers must be within 0 - 254 range it is not possible to find an id within that range that points invalid table

5.3.11. unsupported instruction not specified

Informs user that test was skipped as any instruction from the set defined by openflow spec was specified using *unsupported_instruction_id* configuration parameter. At least one instruction must be specified to perform particular test

5.3.12. only single table supported

Informs user that test was skipped as only single table is supported by the DUT while at least two tables must be supported to perform the test

5.3.13. all metadata write bits supported

Informs user that test was skipped as all metadata bits can be written by the particular table while at least one bit that cannot be written by the table is needed to perform the test.

5.3.14. arbitrary masks supported

Informs user that test was skipped as DUT switch support for arbitrary bitmasks on specific match fields by setting up *arbitrary_masks_supported* configuration parameter to True

5.3.15. unsupported masked match type not specified

Informs user that test was skipped as any masked match type from the set defined by openflow spec was specified using *unsupported_masked_match_id* configuration parameter. At least one masked match type must be specified to perform particular test

5.3.16. unavailable openflow port id was not specified

Informs user that test was skipped as any port id from the set of OpenFlow ports was specified using *unavailable_of_port* configuration parameter. At least one OpenFlow port must be specified to perform particular test

5.3.17. group action not supported

Informs user that test was skipped as according to the *groups_supported* configuration parameter group action is not supported

5.3.18. set_field action not supported

Informs user that test was skipped as according to the *set_field_supported* configuration parameter group action is not supported

5.3.19. meters action not supported

Informs user that test was skipped as meters action is not supported by the DUT

5.3.20. omitted due to the specification issues

Informs user that test was skipped due to the wrong test specification

5.3.21. default table-miss behavior config not available

Informs user that test was skipped as according to the *table_miss_config_supported* configuration parameter table-miss behaviour (taking place when packet is not matching any flows in the table) is not supported

5.3.22. immediate multipart replies detected

Informs user that test was skipped as DUT replies to each multipart request immediately.

5.3.23. insufficient number of tables

Informs user that test was skipped as DUT do not support enough tables to perform the test

5.3.24. performance tests disabled

Informs user that particular test is a performance test and was skipped because *performance_tests_allowed* configuration parameter was set to *False*

5.3.25. optional counter not supported

Informs user that test was skipped as DUT do not support particular optional counter to which test refers

5.3.26. buffers not supported

Informs user that test was skipped as according to the *buffers_supported* configuration parameter group buffers are not supported

5.3.27. unsupported buffer timeout value

Informs user that test was skipped as buffer timeout value (*buffer_timeout* configuration parameter) is too small while test specification is designed for specific timeout values.

5.3.28. specified tunnel type not supported

Informs user that test was skipped as tunnel type prefered for the test (*preferred_410_test_tunnel_type* configuration parameter) is not supported by the DUT

5.3.29. experimenter parameters not specified

Informs user that test was skipped as id of experimenter unsupported by the DUT was not provided by setting *unsupported_experimenter_id* configuration parameter

5.3.30. required field not found

Informs user that test was skipped due to the one of the following reasons: (1) as DUT supports all the fields in the set_field context and at least one unsupported field is obligatory to perform the test. Or (2) as none of the oxms available for set_field was found while at least one oxm is needed to perform the test.

5.3.31. could not generate unique mac address

Informs user that test was skipped as system was not able to find unique mac address that was not reported by any port of the DUT

5.3.32. expected error could not be triggered

In case when expected error could not be triggered informs user that test was skipped due to the one of the following reasons: (1) number of attempts to add a meter has not exceeded max meter capacity. Number of attempts is limited by the *meter_add_max_threshold* configuration parameter, while max meter capacity is defined by the *meter_max_n* configuration parameter. Thus, if *meter_add_max_threshold* < *meter_max_n* meter capacity was not reached and result is not reliable. Or (2) max meter was not configured using *meter_max_n* configuration paramterer.

5.3.33. table features config not supported

Informs user that test was skipped as DUT do not support configuration of table features. Support for table features configuration is indicated by the *table_features_config_supported* configuration parameter.

5.3.34. table features config not enabled

Informs user that test was skipped as configuration of table features was not enabled on the DUT. Table features must be enabled to perform the test. User should refer to the DUT manual to enable/disable table features configuration.

5.3.35. logical interfaces not supported

Informs user that test was skipped as DUT do not support logical interfaces features. Support for logical interfaces is indicated by the *logical_interfaces_supported* configuration parameter.

5.3.36. test for exclusive importance eviction mode

Informs user that test was skipped as importance flag in eviction properties in OpenFlow table description is not the only set flag. While test is developed to test only importance mode of eviction mechanism.

5.3.37. test for exclusive lifetime eviction mode

Informs user that test was skipped as lifetime flag in eviction properties in OpenFlow table description is not the only set flag. While test is developed to test only lifetime mode of eviction mechanism.

5.3.38. eviction not supported

Informs user that test was skipped as DUT do not support eviction mechanism. Support for eviction mechanism is obtained from the table description. User may refer to the DUT manual in order to verify if DUT is claimed to support eviction mechanism.

5.3.39. eviction properties could not be modified

Informs user that test was skipped as system was not able to modify eviction properties. Modification of properties is obligatory to perform the test.

5.3.40. bundles not supported

Informs user that test was skipped as DUT do not support bundles. Support for bundles is indicated by the *bundles_supported* configuration parameter.

5.3.41. meter mod failed

Informs user that test was skipped as system was not able to modify meter (e.g. add new meter). Modification of meter is obligatory to perform the test.

5.3.42. flow_mod failed

Informs user that test was skipped as system was not able to modify flow (e.g. add new flow). Modification of flow is obligatory to perform the test.

5.3.43. egress tables not supported

Informs user that test was skipped as DUT do not support egress tables.

5.3.44. insufficient number of egress tables

Informs user that test was skipped as DUT do not support minimum number of egress tables. To perform the test support for more than one egress table is obligatory.

5.3.45. schedule thresholds cannot be set

Informs user that test was skipped as system was not able to set thresholds related to the scheduling features. Setting the thresholds is obligatory to perform the test.

5.3.46. test result cannot be reliably verified

Informs user that test was skipped as system is not able to verify the result of the test. In some cases particular test can be verified only if one of the previous tests passed. This message is generated when the previous (obligatory) test failed.

5.3.47. tests filtered by openflow version

Informs user that test was skipped as specified OpenFlow version tests are disabled.

5.4. System errors

5.4.1. dataplane connection failed

Informs user that problem with *dataplane* connection occurred. Please check *dataplane* configuration and *dataplane* availability.

5.4.2. controller connection failed

Informs user that problem with *controller* connection occurred. Please check *controller* configuration and *dataplane* availability.

5.4.3. no dataplanes discovered

Informs user that none of the dataplanes was discovered by the system. At least one *dataplane* must be configured.

5.4.4. no controller discovered

Informs user that none of the network *controllers* was discovered by the system. At least one *controller* must be configured.

5.4.5. specified dpid not connected

Informs user that system specified dpid is not available from the system perspective. Please check provided dpid, verify connection, configuration and availability.

5.4.6. no available dpid discovered

Informs user that none of the dpid is available from the system perspective In case when any dpid was provided by the user. Please verify connection, configuration and availability. User may also try to provide dpid.

5.4.7. not enough interfaces specified

Informs user that error occured because system requires at least two configured interfaces. Please check the configuration.

5.4.8. dataplane ntp sync error

Informs user that *dataplane* is not able to synchronize with ntp server.

5.4.9. controller ntp sync error

Informs user that *controller* is not able to synchronize with ntp server.

5.4.10. unsupported fail mode

Informs user that particular test failed because value of *fail_mode* configuration parameter is invalid.

5.5. System prompts

Describe all the prompts that sistem can generate. For each prompt describe possible users answers, their meaning, and possible results of bad configuration. Describe and clearly explain all the actions that may be needed to be done by the system user.

5.5.1. REBOOT DUT

Informs user that DUT should be rebooted. Both options are possible: to unplug power cable or to make DUT to reboot using its user interface.

5.5.2. CONFIGURE DUT TO CONNECT TO PORT *port_number*, SSL *ENABLED/DISABLED*

Informs user that DUT should be configured in order to establish control channel with network controller. Expected port number is provided as well as information if SSL is enabled or disabled.

5.5.3. RESTORE PORT TO UP STATE…

Informs user that OpenFlow port that was previously set down should be enabled again

5.5.4. DISCONNECT PORT *port_number* CABLE OR PUT THE PORT DOWN…

Informs user that particular OpenFlow port sholud be set down or cable should be removed from the port

5.5.5. CONFIGURE DEFAULT TABLE-MISS BEHAVIOR TO PACKET-IN…

Informs user to set up default table-miss DUT behaviour to generate corresponding packet-in message. User should refer to the DUT manual in order to configure default table-miss

5.5.6. CONFIGURE DEFAULT TABLE-MISS BEHAVIOR TO DROP…

Informs user to set up default table-miss DUT behaviour to drop corresponding packet. User should refer to the DUT manual in order to configure default table-miss

5.5.7. PROVIDE ID OF A CONFIGURED QUEUE OR SETUP QUEUE ON A SELECTED PORT AND ENTER ITS ID:

Asks user to provide id of the queue configured on the particular OpenFlow port. In case when queue has not been configured yet user should configure it first. User should refer to the DUT manual in order to configure queues

5.5.8. PROVIDE ID OF A PORT WITH CONFIGURED QUEUES OR SETUP QUEUES ON A SELECTED PORT AND ENTER ITS ID:

Asks user to provide id of the OpenFlow port with queue configured. In case when queue has not been configured yet user should configure it first. User should refer to the DUT manual in order to configure queues

5.5.9. REMOVE QUEUES CONFIGURED DURING TEST…

Asks user to remove any configuration of queues. User should refer to the DUT manual in order to remove queues configuration

5.5.10. SPECIFY EXPECTED CURR_FEATURES FOR PORT *port_number*:

Asks user to define expected current features in a description of particular standard port of the DUT. Value will be assigned to the curr_features configuration parameter and user will be prompted for the value only if curr_features has not been configured using configuration file. User should refer to the DUT manual in order to provide proper value.

5.5.11. SPECIFY EXPECTED ADV_FEATURES FOR PORT *port_number*:

Asks user to define expected features being advertised by the port in a description of particular standard port of the DUT. Value will be assigned to the *adv_features* configuration parameter and user will be prompted for the value only if *adv_features* has not been configured using configuration file. User should refer to the DUT manual in order to provide proper value.

5.5.12. SPECIFY EXPECTED SUPP_FEATURES FOR PORT *port_number*:

Asks user to define expected features supported by the port in a description of particular standard port of the DUT. Value will be assigned to the *supp_features* configuration parameter and user will be prompted for the value only if *supp_features* has not been configured using configuration file. User should refer to the DUT manual in order to provide proper value.

5.5.13. SPECIFY EXPECTED PEER_FEATURES FOR PORT *port_number*:

Asks user to define expected features being advertised by the peer in a description of particular standard port of the DUT. Value will be assigned to the *peer_features* configuration parameter and user will be prompted for the value only if *peer_features* has not been configured using configuration file. User should refer to the DUT manual in order to provide proper value.

5.5.14. SPECIFY EXPECTED MAX_SPEED FOR PORT *port_number*:

Asks user to define expected maximum port bitrate in kbps according to the description of particular standard port of the DUT. Value will be assigned to the *max_speed* configuration parameter and user will be prompted for the value only if *max_speed* has not been configured using configuration file. User should refer to the DUT manual in order to provide proper value.

5.5.15. SPECIFY EXPECTED CURR_SPEED FOR PORT *port_number*:

Asks user to define expected current port bitrate in kbps according to the description of particular standard port of the DUT. Value will be assigned to the *curr_speed* configuration parameter and user will be prompted for the value only if *curr_speed* has not been configured using configuration file. User should refer to the DUT manual in order to provide proper value.

5.5.16. PROVIDE NUMBER OF QUEUES CONFIGURED ON PORT *port_number*:

Asks user to provide number of queues configured on particular OpenFlow port. User should refer to the DUT manual in order to configure queues and provide proper value.

5.5.17. PROVIDE TOTAL NUMBER OF QUEUES CONFIGURED ON A DEVICE:

Asks user to provide number of queues configured on the DUT. User should refer to the DUT manual in order to configure queues and provide proper value.

5.5.18. PROVIDE ID OF A PORT WITHOUT ANY QUEUES CONFIGURED:

Asks user to provide OpenFlow port identifier that has not any queue configured. User should refer to the DUT manual in order to configure queues and provide proper value.

5.5.19. CONFIGURE FORWARDING TABLE: *destination_mac* AVAILABLE ON PORT *DUT_port_id*

Asks user to add entry to the forwarding table in order to direct frames with provided destination mac address to the selected output port of the DUT. Make sure to provide expiration time sufficient (long enough) to perform the test. User should refer to the DUT manual in order to configure forwarding tables.

5.5.20. CREATE SEPARATE VLAN *vlan_id* EXCLUDED FROM OPENFLOW INSTANCE…

Asks user to create a vlan with specified id and not being a part of OpenFlow instance within DUT.

5.5.21. REMOVE PREVIOUSLY CREATED VLAN…

Asks user to clear clear VLAN configuration in the DUT. User should refer to the DUT manual in order to get familiar with VLAN configuration

5.5.22. MOVE INTERFACES *interface_id_1* AND *interface_id_2* TO VLAN *vlan_id* (TAGGED)...

Asks user to assign particular OpenFlow interfaces to the vlan instance with provided id and being excluded from the OpenFlow instance. Configured ports must send tagged frames. User should refer to the DUT manual in order to get familiar with assigning ports to the VLANs.

5.5.23. RE-ADD PORTS TO OPENFLOW INSTANCE…

Asks user to restore port configuration and assign ports back to the OpenFlow instance instead of vlan excluded from OpenFlow. User should refer to the DUT manual in order to get familiar with assigning ports to the VLANs.

5.5.24. PROVIDE THE NAME OF OPENFLOW PORT *port_id*:

Asks user to provide the name of the OpenFlow port indicated by the id.

5.5.25. CONFIGURE *tunnel_type* TUNNEL (REMOTE_IP=ANY) AND PROVIDE TUNNEL KEY:

Asks user to configure in DUT tunnel of particular type (e.g. VxLAN, MPLS, GRE) with any remote ip. After configuration user must provide key of the configured terminal (e.g. VNI, mpls label, GRE tunnel id). User should refer to the DUT manual in order to get familiar with tunnels configuration.

5.5.26. CONFIGURE *tunnel_type* TUNNEL (REMOTE_IP=UNIQUE PRIVATE SUBNET ADDRESS, KEY=ANY VALUE)...

Asks user to configure in DUT tunnel of particular type (e.g. VxLAN, MPLS, GRE) with remote ip set to unique private subnet address and any key. User should refer to the DUT manual in order to get familiar with tunnels configuration.

5.5.27. PROVIDE TUNNEL LOGICAL PORT ID:

Asks user to provide id of the logical port associated with the configured tunnel.

5.5.28. PROVIDE TUNNEL KEY:

Asks user to provide key of the configured tunnel.

5.5.29. PROVIDE TUNNEL REMOTE IP:

Asks user to provide remote IP of the configured tunnel.

5.5.30. REMOVE PORT *port_id* (*interface_name*) FROM OPENFLOW INSTANCE…

Asks user to exclude OpenFlow port from the OpenFlow instance. Port is indicated using port identifier and described by the assigned name. User should refer to the DUT manual in order to get familiar with excluding OpenFlow ports from the OpenFlow instance.

5.5.31. CONFIGURE PORT *port_id* (*interface_name*) WITH UNIQUE IP ADDRESS OR ADD TO A MANAGEMENT VLAN…

Asks user to either configure unique ip address at the particular DUT port or add that port to the management VLAN. Port is indicated using port identifier and described by the assigned name. User should refer to the DUT manual in order to get familiar with IP address configuration as well as configuring VLANs and assigning interfaces to them.

5.5.32. CONFIGURE INTERFACE *interface_name* WITH UNIQUE IP ADDRESS FROM THE SAME SUBNET AS *tunnel_remote_ip*…

Asks user configure ip address at the particular DUT port. The IP address should include in the subnet configured as a remote ip of the tunnel. Port is indicated using the assigned name. User should refer to the DUT manual in order to get familiar with IP address configuration.

5.5.33. PROVIDE CONFIGURED PORT *port_id* (*interface_name*) IP ADDRESS (IP/MASK):

Asks user to provide the IP address and mask of the configured IP port. Expected format is IP.IP.IP.IP/M. Port is indicated using port identifier and described by the assigned name.

5.5.34. REMOVE ADDRESS FROM PREVIOUSLY CONFIGURED INTERFACE…

Asks user to clear ip address configuration of the port configured in the context of the test. User should refer to the DUT manual in order to get familiar with IP address configuration.

5.5.35. SWITCH DUT OPENFLOW VERSION TO {*ver_id*} (*ver_name*), RESET CONNECTION AND PRESS ENTER TO PROCEED…

Asks user to change DUT OpenFlow version to the particular one defined by the version identifier and version name. After change connection must be reseted. User should refer to the DUT manual in order to change OpenFlow datapath version.

5.5.36. RESET DATAPATH CONNECTION…

Asks user to reset all datapath connections. User should refer to the DUT manual in order to reset datapath connections.

5.6. Example of system output

Below user can find an excerpt of system output to present described system outputs. Thus, mostly tests with skip reason provided are presented in the output. “OUTPUT CUT” indicates points of the output when original output was cut. Complete output can be found at http://download.netrsr.com/~micrze/tlog_28102016.

Test names are designed according to the following scheme
test_[nvct/onf]_[013/014/015]_*testCaseID_testID_testDescp*, where
nvct/onf - indicates if test originates from the onf test specification or nvct test specification (for details refer to the Specification Section)
013/014/015 - indicates what version of OpenFlow protocol is tested (1.3 or 1.4 or 1.5, respectively)
testCaseID - numerical test case identifier as tests are grouped into test cases
testID - numerical identifier of test
testDesc - meaningful description and name of the test

Test cases names are designed according to the following scheme:
[nvct/onf]_[013/014/015]_*testCaseID_testCaseDesc.testSubCaseDesc,* where
all the components described for test names remain unchanged and
testCaseDesc - descriptive and meaningful name of the test case
testSubCaseDesc - descriptive and meaningful name of the test sub case, as test cases are further divided into sub cases

At the beginning of the system output presents results of the tests interactively during the execution. The scheme is as follows: *testName* (*testCaseName)* … ok/FAIL/skipped *reason*
where,
testName and *testCaseName* have been already explained.
ok - means that the test was successfully passed
FAIL - denotes that DUT failed passing the test
skipped *reason* - informas that test was skipped due to the provided *reason* (for reason explanation please refer to the System messages Section

Once all tests are executed NVCT provides summary results. First, all successfully passed tests are listed with “SUCCESS:” label. For some of them additional note is provided, “DATA AVAILABLE” denotes that numerical results were obtained for that test and are provided in later parts of the output. Those numerical results are provided with “DATA:” label. For information about the particular numerical values please refer to the corresponding test in Tests Specification Section. Next, all skipped tests are listed with “SKIPPED:” label, for each test skip reason is provided. Finally, all tests that DUT failed to pass are listed with “FAIL:”

label. At the very end, total number of tests performed is provided along with execution time and information about number of failed and skipped tests.

```
test_nvct_013_010_010_normal_forwarding_predefined_rule (nvct_013_010_hybrid_mode.BasicL2TestCase) ... skipped
'manual tests disabled'
=============> OUTPUT CUT ⇦===============
----------
SWITCH DATAPATH OF VERSION TO 0x04 (1.3), RESET CONNECTION AND PRESS ENTER TO PROCEED...FAIL
test_nvct_013_010_030_normal_forwarding_unicast_flood (nvct_013_010_hybrid_mode.BasicL2TestCase) ... ok
test_nvct_013_010_040_action_flood (nvct_013_010_hybrid_mode.BasicL2TestCase) ... ok
test_nvct_013_010_110_vlan_isolation_forwarding_predefined_rule (nvct_013_010_hybrid_mode.VlanIsolationTestCase) ...
skipped 'manual tests disabled'
test_nvct_013_020_110_vxlan_set_field (nvct_013_020_tunneling.TunnelSetupTwoTestCase) ... skipped 'manual tests
disabled'
test_nvct_013_040_030_capacity_l2_header_fields (nvct_013_040_performance.FlowTableCapacityTestCase) ... skipped 'test
skipped due to estimated long execution time'
=============> OUTPUT CUT ⇦===============
----------
SWITCH DATAPATH OF VERSION TO 0x05 (1.4), RESET CONNECTION AND PRESS ENTER TO PROCEED...ok
test_nvct_014_490_020_eviction_properties_modification (nvct_014_490_of14.EvictionTestCase) ... FAIL
test_nvct_014_490_030_flow_importance_field (nvct_014_490_of14.EvictionTestCase) ... ok
test_nvct_014_490_110_eviction_disabled_table_full_error (nvct_014_490_of14.EvictionTestCase) ... skipped 'test skipped
due to estimated long execution time'
test_nvct_014_490_130_eviction_flags_importance (nvct_014_490_of14.EvictionTestCase) ... skipped 'test for importance
eviction mode only'
test_nvct_014_490_140_eviction_flags_lifetime (nvct_014_490_of14.EvictionTestCase) ... skipped 'test for lifetime eviction
mode only'
test_nvct_014_490_150_eviction_flags_importance_mod (nvct_014_490_of14.EvictionTestCase) ... skipped 'eviction
properties could not be modified'
test_nvct_014_490_310_flow_removed_reason_meter_delete (nvct_014_490_of14.MeterModTestCase) ... skipped 'meter mod
failed'
=============> OUTPUT CUT ⇦===============
----------
SWITCH DATAPATH OF VERSION TO 0x06 (1.5), RESET CONNECTION AND PRESS ENTER TO PROCEED...FAIL
test_nvct_015_490_020_egress_table_0_table_verification (nvct_015_490_of15.TablePipelineTestCase) ... skipped 'egress
tables not supported'
test_nvct_015_490_510_oxm_of_tcp_flags_ack (nvct_015_490_of15.OxmTcpFlagsTestCase) ... ok
test_nvct_015_490_550_oxm_of_tcp_flags_fin (nvct_015_490_of15.OxmTcpFlagsTestCase) ... ok
test_nvct_015_490_610_scheduled_bundle_correct_usage (nvct_015_490_of15.ScheduledBundleTestCase) ... skipped
'schedule thresholds cannot be set'
test_nvct_015_490_730_copy_header_to_header (nvct_015_490_of15.CopyActionTestCase) ... ok
=============> OUTPUT CUT ⇦===============
----------
SWITCH DATAPATH OF VERSION TO 0x04 (1.3), RESET CONNECTION AND PRESS ENTER TO PROCEED...ok
test_onf_013_010_120_fail_standalone_mode (onf_013_010_control_channel.FailureModeTestCase) ... skipped 'standalone
mode only test'
test_onf_013_010_130_entries_stay_active (onf_013_010_control_channel.FailureModeTestCase) ... skipped 'test skipped due
to estimated long execution time'
test_onf_013_040_010_datapath_id (onf_013_040_controller_to_switch_messages.SwitchFeaturesTestCase) ... ok
test_onf_013_040_090_reassemble_ip_fragments (onf_013_040_controller_to_switch_messages.SwitchFeaturesTestCase) ...
FAIL
test_onf_013_150_070_unsupported_meta_data (onf_013_150_flow_table_errors.FlowTableErrorsTestCase) ... skipped 'all
metadata write bits supported'
test_onf_013_150_110_bad_error_mask (onf_013_150_flow_table_errors.FlowTableErrorsTestCase) ... skipped 'arbitrary
masks supported'
test_onf_013_150_160_currently_invalid_port (onf_013_150_flow_table_errors.FlowTableErrorsTestCase) ... skipped
'unavailable of port id was not specified'
test_onf_013_180_410_duration_precision (onf_013_180_counters.CountersTestCase) ... skipped 'test dependency
requirements not met'
```

(onf_013_250_switch_config_protocol_message.SwitchConfigProtocolMessageTestCase) ... skipped 'default table-miss behavior config not available'
(onf_013_260_flow_mod_protocol_messages.FlowModProtocolMessageTestCase) ... ok
test_onf_013_340_130_received_frame_errors (onf_013_340_multipart_reply_four.PortStatsTestCase) ... skipped 'test specification omitted'
test_onf_013_340_270_current_features (onf_013_340_multipart_reply_four.PortDescTestCase) ... skipped 'expected value not configured'
=============> OUTPUT CUT <===============
==
SUCCESS: test_nvct_013_010_030_normal_forwarding_unicast_flood (nvct_013_010_hybrid_mode.BasicL2TestCase)
SUCCESS: test_nvct_013_010_040_action_flood (nvct_013_010_hybrid_mode.BasicL2TestCase)
SUCCESS: test_nvct_013_040_120_flow_add_rate (nvct_013_040_performance.FlowModPerformanceTestCase) DATA AVAILABLE
SUCCESS: test_nvct_013_040_130_wildcard_flow_add_rate (nvct_013_040_performance.FlowModPerformanceTestCase) DATA AVAILABLE
(onf_013_140_flow_table_modifications.FlowTableModificationsTestCase)
=============> OUTPUT CUT <===============

==
DATA: test_nvct_013_040_110_single_flow_add_latency (nvct_013_040_performance.FlowModPerformanceTestCase)
mean timestamp diff: 0.030780 (5 samples)
--
DATA: test_nvct_013_040_115_single_flow_add_latency_occupied_table
(nvct_013_040_performance.FlowModPerformanceTestCase)
0: 0.0242425918579
100: 0.0219265937805
500: 0.0212105751038
2000: 0.019159412384
5000: 0.0210085391998
--
DATA: test_nvct_013_040_120_flow_add_rate (nvct_013_040_performance.FlowModPerformanceTestCase)
mean flow add rate: 1056.095275 [flows/sec] (5 samples)
--
DATA: test_nvct_013_040_130_wildcard_flow_add_rate (nvct_013_040_performance.FlowModPerformanceTestCase)
mean wildcarded flow add rate: 1288.767434 [flows/sec] (5 samples)
--
DATA: test_nvct_013_040_145_single_flow_modification_latency_occupied_table
(nvct_013_040_performance.FlowModPerformanceTestCase)
0: 0.0268593788147
100: 0.0285471439362
500: 0.0268730163574
2000: 0.0263291358948
5000: 0.031657743454
--
DATA: test_nvct_013_040_150_flow_modification_rate (nvct_013_040_performance.FlowModPerformanceTestCase)
mean flow modification rate: 599.253995 [flows/sec] (5 samples)
--
DATA: test_nvct_013_040_160_single_flow_delete_latency (nvct_013_040_performance.FlowModPerformanceTestCase)
mean timestamp diff: 0.028311 (5 samples)
--
DATA: test_nvct_013_040_165_single_flow_delete_latency_occupied_table
(nvct_013_040_performance.FlowModPerformanceTestCase)
0: 0.0286270141602
100: 0.0242292404175
500: 0.0264571666718
2000: 0.0304738998413
5000: 0.0312313556671
--
DATA: test_nvct_013_040_170_flow_delete_rate (nvct_013_040_performance.FlowModPerformanceTestCase)
mean flow delete rate: 953.932075 [flows/sec] (5 samples)
--
DATA: test_nvct_013_040_210_packet_out_handling (nvct_013_040_performance.PacketOutPerformanceTestCase)

```
mean timestamp diff: 0.031632 (5 samples)
--
DATA: test_nvct_013_040_220_packet_out_size (nvct_013_040_performance.PacketOutPerformanceTestCase)
latency diff (64->128): -7.74%
latency diff (128->256): -13.44%
latency diff (256->512): -0.36%
latency diff (512->1024): -8.25%
latency diff (1024->1280): 47.95%
--
=============> OUTPUT CUT ⇦===============
======================================================================
SKIPPED: test_nvct_013_010_010_normal_forwarding_predefined_rule (nvct_013_010_hybrid_mode.BasicL2TestCase):
manual tests disabled
SKIPPED: test_nvct_013_010_110_vlan_isolation_forwarding_predefined_rule
(nvct_013_010_hybrid_mode.VlanIsolationTestCase): manual tests disabled
(onf_013_140_flow_table_modifications.FlowTableModificationsTestCase): test dependency requirements not met
=============> OUTPUT CUT ⇦===============
======================================================================
FAIL: test_onf_013_440_280_port_mod_failed_bad_config
(onf_013_440_error_message_two.RegularErrorMessageSectionTwoTestCase)
FAIL: test_onf_013_440_290_port_mod_failed_bad_advertise
(onf_013_440_error_message_two.RegularErrorMessageSectionTwoTestCase)
=============> OUTPUT CUT ⇦===============
----------------------------------------------------------------------
Ran 466 tests in 2796.814s

FAILED (failures=72, skipped=111)
```

6. Specifications

The main contribution of this section is to provide specification of tests designed for the NVCT system in order to extend functionalities coverage ratio. All the tests available in freeware version were carefully described providing for each test: purpose, methodology, references to OpenFlow Protocol specification, expected connections topology, possible test results, recommendations and additional remarks. Tests available in the full version are only listed. Additionally, technical details are described at the very end of the section.

6.1. Tests Specification

NVCT.SuiteNumber - ChapterTitle

Short description

Remarks
Problems related to this chapter

NVCT.SuiteNumber.TestNumber - TestCaseTitle

Purpose
Goal of the test

Methodology
Main description of the test

Specification text (Optional - only for testing features specified in OpenFlow)
OFP specification text relating the test case

Topology
Required topology

Results
Possible results

Test recommendations (Optional)
Useful informations, for example suggested values

Additional remarks (Optional)
Explanations such as ambiguities

NVCT.10 - L2/OF hybrid mode

The L2/OF hybrid mode test suit verifies the ability of the DUT to work as a legacy Ethernet switch.

Remarks
Hybrid mode can be defined in two ways:

- mode separation based on action NORMAL (action NORMAL means to handle packet utilizing L2 switching capabilities) or packet classification mechanisms (approach presented in OpenFlow specification)
- mode separation based on VLANs (VLAN isolation), selected VLANs are handled utilizing OpenFlow, other VLANs are switched utilizing L2 switching capabilities of DUT (approach presented by vendors)

Both approaches may be combined as match can be performed based on the VLAN ID.

The later approach should be default (as indicated in OpenFlow specification). However, test operator should be able to change approach to the first one.

For tests NVCT.10.10-40 default behaviour is tested (action NORMAL). For tests NVCT.10.110-140 VLAN isolation is tested (make sure that changing the default behaviour is a part of the test).

NVCT.10.10 - action NORMAL - forwarding packet based on predefined L2 rule

NVCT.10.20 - action NORMAL - forwarding packet based on L2 rule (successful MAC address learning)

NVCT.10.30 - action NORMAL - dropping packet based on L2 rule (flooding as a part of MAC address learning process)

NVCT.10.40 - action FLOOD - forwarding packet to all standard ports

NVCT.10.110 - VLAN isolation - forwarding packet based on predefined L2 rule

NVCT.10.120 - VLAN isolation - forwarding packet based on predefined L2 rule (successful MAC address learning)

NVCT.10.130 - VLAN isolation - forwarding packet based on predefined L2 rule (flooding as a part of MAC address learning process)

NVCT.10.140 - VLAN isolation - forwarding packets based on predefined L2 rule or OpenFlow pipeline

NVCT.20 - Tunnelling

The tunnelling support test suite verifies the ability of the DUT to support encapsulation utilizing logical port abstraction.

Remarks
Packets associated with logical ports may have an extra field called Tunnel-ID which carries additional metadata regarding that port. Value of the Tunnel-ID is usually extracted from the header, may be used in pipeline processing and finally may be multiplexed in the outgoing header.

The mapping of the optional encapsulation metadata in the Tunnel ID field is defined by the logical port implementation, it is dependant on the type of logical port and it is implementation specific. Following recommendations are provided in OpenFlow Switch Specification 1.3.4 (ch. 7.2.3.9; pg. 63): We recommend that for a packet received via a GRE tunnel including a (32-bit) key, the key is stored in the lower 32-bits and the high bits are zeroed. We recommend that for a MPLS logical port, the lower 20 bits represent the MPLS Label. We recommend that for a VxLAN logical port, the lower 24 bits represent the VNI.

NVCT.20.10 - Tunnel-ID extracting from header - GRE tunnel

NVCT.20.20 - Tunnel-ID extracting from header - MPLS logical port (DEPRECATED)

NVCT.20.30 - Tunnel-ID extracting from header - VxLAN logical port

NVCT.20.40 - Tunnel-ID extracting from header - physical port

NVCT.20.100 - VxLAN based forwarding

NVCT.20.110 - VxLAN - set-field action

NVCT.30 - Security features

The security test suite verifies safe communication between DUT and OpenFlow controller as well as ability of DUT to defend against DoS attacks.

Remarks
Some initial tests related to the secure communication between DUT and controller are provided in ONF Conformance Test Specification v 1.0 for OpenFlow Switch. Tests developed in NVCT aim at extending those tests and verifying additional features.

NVCT.30.10 - TLS with default TCP port - encrypted controller-to-switch message

NVCT.30.20 - TLS with default TCP port - encrypted asynchronous message

NVCT.30.30 - TLS with default TCP port - encrypted symmetric message

NVCT.30.40 - TLS with non default TCP port - encrypted controller-to-switch message

NVCT.30.50 - TLS with non default TCP port - encrypted asynchronous message

NVCT.30.60 - TLS with non default TCP port - encrypted symmetric message

NVCT.30.70 - TLS session timeout

NVCT.30.80 - Fail standalone mode behavior

NVCT.30.90 - Alternate encryption configuration (skipped due to the lack of support for IPSec and VPN in available SDN controllers)

NVCT.30.100 - Mechanisms against the DoS attack (skipped due to the problems with heavy traffic generation)

NVCT.40 - Performance

The performance test suit assess the efficiency of DUT while installing and forwarding new flows as well as forwarding, modifying and removing existing flows.

Remarks
It must be noted that the main purpose of the test suite is to measure performance of crucial OpenFlow related features. Among the others questions, test suite will for example answer the following: how fast switch is able to process messages received from the controller, how fast new entries are installed in forwarding table or how fast modifications to the existing entries are introduced.
Therefore, simple packet forwarding performance is not a main issue as after having entries installed in forwarding table packets should be forwarded at line rate which can be verified utilizing standardized performance tests.

NVCT.40.10 - Number of Forwarding Tables

NVCT.40.20 - Flow Table Capacity - all header fields

NVCT.40.30 - Forwarding Table Capacity - layer 2 fields

NVCT.40.40 - Forwarding Table Capacity - layer 3 fields

NVCT.40.110 - Flow-mod performance - single flow add latency

NVCT.40.115 - Flow-mod performance - single flow add latency in occupied table

NVCT.40.120 - Flow-mod performance - flow add rate

NVCT.40.130 - Flow-mod performance - wildcard flow add rate

NVCT.40.140 - Flow-mod performance - single flow modification latency

NVCT.40.145 - Flow-mod performance - single flow modify latency in occupied table

NVCT.40.150 - Flow-mod performance - flow modification rate

NVCT.40.160 - Flow-mod performance - single flow delete latency

NVCT.40.165 - Flow-mod performance - single flow delete latency in occupied table

NVCT.40.170 - Flow-mod performance - flow delete rate

NVCT.40.210 - Packet In/Out performance - Packet Out handling (deprecated due to the problems with heavy traffic generation)

NVCT.40.220 - Packet In/Out performance - Packet size (deprecated due to the problems with heavy traffic generation)

NVCT.40.230 - Packet In/Out performance - Buffer size (skipped due to the problems with heavy traffic generation)

NVCT.40.310 - Table-miss performance - default behaviour

NVCT.40.320 - Table-miss performance - maximum throughput (skipped due to the problems with heavy traffic generation)

NVCT.40.410 - Statistics - impact of tracking on flow add message (skipped due to the lack of support statistics switching off in OpenFlow switches)

NVCT.40.420 - Statistics - impact of tracking on traffic handling (skipped due to the lack of support statistics switching off in OpenFlow switches)

NVCT.40.430 - Statistics - impact of polling on traffic handling

NVCT.40.510 - QoS - rate-limiting (DEPRECATED due to the problems with heavy traffic generation)

NVCT.50 - Multiple Tables Features

The Multiple Tables Features test suite verifies the DUT behaviour in the context of multi-table pipeline processing.

Remarks
Multi-table processing allows to optimize the most sophisticated rules being configured in the controller. As a result flow processing time may be significantly reduced in comparison to the case with only single table. Requirements for devices supporting multiple tables are not covered in ONF Conformance Test Specification v 1.0 for OpenFlow Switch.

Pipeline processing instructions allow packets to be sent to subsequent tables for further processing and allow information, in the form of metadata, to be communicated between tables. Table pipeline processing stops when the instruction set associated with a matching flow entry does not specify a next table; at this point the packet is usually modified and forwarded (see 5.10).
OpenFlow Switch Specification 1.3.4 (ch.2; pg. 8)

Based on the test 40.30 from ONF Conformance Test Specification v 1.0 for OpenFlow Switch number of supported tables should be found for the DUT. Number of supported tables will be referred as n_tables in this test suite. In case when only one table is supported all the tests in this test suite should be marked not-applicable.

NVCT.50.10 - Multiple table pipeline - Table 0

NVCT.50.20 - Multiple table pipeline - Table n>0

NVCT.50.30 - Multiple table pipeline - Table n>0 default action

NVCT.50.40 - Multiple table pipeline - Skip tables

NVCT.50.50 - Multiple table pipeline - Goto-Table instruction next-table-id < table-id

NVCT.50.60 - Multiple table pipeline - Goto-Table instruction in last table

NVCT.50.70 - Multiple table pipeline - Table-miss chain

NVCT.60 - Open Flow Protocol v. 1.4 features

The Open Flow Protocol v. 1.4 test suite verifies the DUT behaviour in the context of the most important features introduced in version 1.4 of OFP.

Remarks
The most valuable features introduced in version 1.4 of Open Flow Protocol were selected. The aim is to verify if DUT supports those emerging features. Such a approach is reasonable as only few switches are fully compatible with OFP v1.4 while some of them reports to support selected features. Basic reference for all the tests is OpenFlow Switch Specification 1.4.1.

One of the most important mechanisms is Eviction. Eviction adds a mechanism enabling the switch to automatically eliminate entries of lower importance to make space for newer entries. This enables smoother degradation of behaviour when the table is full. (OpenFlow Switch Specification 1.4.1, ch. B.17.7, pg. 223).

Second crucial mechanism are Vacancy events which add a mechanism enabling the controller to get an early warning based on a capacity threshold chosen by the controller. This allows the controller to react in advance and avoid getting the table full.
(OpenFlow Switch Specification 1.4.1, ch. B.17.8, pg. 224).

Third extension regards to the new reason value for the asynchronous message OFPT_FLOW_REMOVED. If the controller has requested to be notified when flow entries time out or are deleted from tables (see 5.5), the datapath does this with the OFPT_FLOW_REMOVED message.
When a meter is deleted on the switch, all the flow entries that use that meter are removed. This is similar to how group operates. The flow entry that is removed may generate a flow-removed message (depending on config). In version 1.3, we did not have a reason for this condition, version 1.4 fixes that bug.
(OpenFlow Switch Specification 1.4.1, ch. 7.4.2, pg. 142, B.17.4, pg. 222).
In addition, as OFPT_FLOW_REMOVED message tests are not present in ONF Conformance Test Specification v 1.0 we provide them in our system.

Also bundles are introduced in 1.4 version.
Bundle mechanism, enabling to apply a group of OpenFlow message as a single operation (EXT-230). This enables the quasi-atomic application of related changes, and to better synchronise changes across a series of switches.
(OpenFlow Switch Specification 1.4.1, ch. B.17.9, pg. 224).

NVCT.60.10 - Eviction - enabling eviction mechanism

NVCT.60.20 - Eviction - eviction properties modification

NVCT.60.30 - Eviction - flow importance field

NVCT.60.110 - Eviction - eviction disabled table full error

NVCT.60.120 - Eviction of entries

NVCT.60.130 - Eviction of entries - eviction flags

NVCT.60.140 - Eviction of entries - eviction flags

NVCT.60.150 - Eviction of entries - eviction flags

NVCT.60.160 - Eviction of entries - eviction flags

NVCT.60.210 - Vacancy - enabling vacancy mechanism

NVCT.60.220 - Vacancy - vacancy down event

NVCT.60.230 - Vacancy - vacancy up event

NVCT.60.240 - Vacancy - bad threshold values

NVCT.60.310 - Flow removed message reason - meter delete

NVCT.60.410 - Bundle messages - correct usage

Purpose
Verify if DUT behave correctly while handling bundle messages.

Methodology
Configure and connect DUT to controller. After control channel establishment, open bundle request using OFPBCT_OPEN_REQUEST. Use couple of OFPT_BUNDLE_ADD_MESSAGE messages to add different flows matching on a named field (under the Pre-requisites for the match). Close bundle request using OFPBCT_CLOSE_REQUEST and commit it using OFPBCT_COMMIT_REQUEST.
At different stages (after OFPT_BUNDLE_ADD_MESSAGE and OFPBCT_CLOSE_REQUEST messages) verify that before commit none of the flows were added to the flow table and that all of them were added after commit.

Specification text
A bundle is a sequence of OpenFlow modification requests from the controller that is applied as a single OpenFlow operation. If all modifications in the bundle succeed, all of the modifications are retained, but if any errors arise, none of the modifications are retained. (...)
A bundle is specified as all messages conveyed in Bundle Add messages with the same bundle_id on a specific controller connection (see 7.3.9.6). Messages conveyed in a Bundle Add message are formatted like regular OpenFlow messages and have the same semantics. The messages included in a bundle are pre-validated as they are stored in the bundle, minimising the probability of errors when the bundle is applied. The application of the message included in the Bundle Add message is postponed to when the bundle is committed.
OpenFlow Switch Specification 1.4.1 (ch. 6.8.1; pg. 53)

Topology
One control plane connection.

Results
Pass / Fail / Not applicable

Additional remarks
The test may be not applicable as bundle is an optional feature and may not be supported by all OpenFlow switches. A switch is not required to accept arbitrary messages in a bundle, a switch may not accept some message types in bundles, and a switch may not allow all combinations of message types to be bundled together (see 7.3.9.6). At a minimum, if a switch supports bundles, it must be able to support a bundle of multiple flow-mods and port-mods in any order.
OpenFlow Switch Specification 1.4.1 (ch. 6.8.1; pg. 53)

NVCT.60.420 - Bundle messages - immediate error

Purpose
Verify if DUT validates every single message conveyed in bundle and reports an error immediately.

Methodology
Configure and connect DUT to controller. After control channel establishment, open bundle request using OFPBCT_OPEN_REQUEST. Use couple of OFPT_BUNDLE_ADD_MESSAGE messages to add different flows matching on a named field (under the Pre-requisites for the match). One of the OFPT_BUNDLE_ADD_MESSAGE messages should generate an error (for example due to the invalid table_id). Close bundle request using OFPBCT_CLOSE_REQUEST and commit it using OFPBCT_COMMIT_REQUEST.
Verify that error was generated after sending an invalid message (use unique xid to match errors to messages). At different stages (after OFPT_BUNDLE_ADD_MESSAGE and OFPBCT_CLOSE_REQUEST messages) verify that before commit none of the flows were added to the flow table and that also none of them was added after commit.

Specification text
The OpenFlow messages part of a bundle must be pre-validated before they are stored in the bundle (see 7.3.9.6). For each message sent by the controller, the switch must verify that the syntax of the message is valid and that all features in the message are supported features, and immediately return an error message if this message can not be validated. The switch may optionally verify resource availability and may commit resources at this time and generate errors. Messages generating errors when added to the bundle are not stored in the bundle and the bundle is unmodified. When the bundle is committed, most errors will have been already detected and reported. One of the message included in the bundle may still fail during commit, for example due to resource availability. In this case, no message part of the bundle is applied and the switch must generate the error message corresponding to the failure (see 7.3.9.8). Messages of a bundle should have unique xid to help matching errors to messages. If none of the messages part of the bundle generate an error message, the switch informs the controller of the successful application of the bundle.
OpenFlow Switch Specification 1.4.1 (ch. 6.8.3; pg. 53)

Topology
One control plane connection.

Results
Pass / Fail / Not applicable

Additional remarks
The test may be not applicable as bundle is an optional feature and may not be supported by all OpenFlow switches.

NVCT.60.430 - Bundle messages - error after commit

Purpose
Verify if DUT validates messages conveyed in bundle during commit and generates errors.

Methodology
Configure and connect DUT to controller. After control channel establishment, open bundle request using OFPBCT_OPEN_REQUEST. Use couple of OFPT_BUNDLE_ADD_MESSAGE messages to add different flows matching on a named field (under the Pre-requisites for the match). Close bundle request using OFPBCT_CLOSE_REQUEST and commit it using OFPBCT_COMMIT_REQUEST.
At different stages (after OFPT_BUNDLE_ADD_MESSAGE and OFPBCT_CLOSE_REQUEST messages) verify that before commit none of the flows were added to the flow table. Verify that after commit error message was generated due to the resource unavailability and none of the entries was added after commit.

Specification text
The OpenFlow messages part of a bundle must be pre-validated before they are stored in the bundle (see 7.3.9.6). For each message sent by the controller, the switch must verify that the syntax of the message is valid and that all features in the message are supported features, and immediately return an error message if this message can not be validated. The switch may optionally verify resource availability and may commit resources at this time and generate errors. Messages generating errors when added to the bundle are not stored in the bundle and the bundle is unmodified. When the bundle is committed, most errors will have been already detected and reported. One of the message included in the bundle may still fail during commit, for example due to resource availability. In this case, no message part of the bundle is applied and the switch must generate the error message corresponding to the failure (see 7.3.9.8). Messages of a bundle should have unique xid to help matching errors to messages. If none of the messages part of the bundle generate an error message, the switch informs the controller of the successful application of the bundle.
OpenFlow Switch Specification 1.4.1 (ch. 6.8.3; pg. 53)

Topology
One control plane connection.

Results
Pass / Fail / Not applicable

Additional remarks
The test may be not applicable as bundle is an optional feature and may not be supported by all OpenFlow switches.
Group features were utilized for the purpose of the test.

NVCT.60.440 - Bundle messages - ordered messages

Purpose
Verify if DUT applies messages strictly in order when OFPBF_ORDERED flag is set.

Methodology
Configure and connect DUT to controller. After control channel establishment, open bundle request using OFPBCT_OPEN_REQUEST. Use couple of OFPT_BUNDLE_ADD_MESSAGE messages to add different flows matching on a named field (under the Pre-requisites for the match) with OFPBF_ORDERED flag set.
Close bundle request using OFPBCT_CLOSE_REQUEST and commit it using OFPBCT_COMMIT_REQUEST.
At different stages (after OFPT_BUNDLE_ADD_MESSAGE and OFPBCT_CLOSE_REQUEST messages) verify that before commit none of the flows were added to the flow table and that all of them were added after commit in the order in which messages were sent.

Specification text
If the flag OFPBF_ORDERED is specified (see 7.3.9.3), the messages part of the bundle must be applied strictly in order, as if separated by OFPT_BARRIER_REQUEST messages (however no OFPT_BARRIER_REPLY is generated). If this flag is not specified, messages don’t need to be applied in order.
OpenFlow Switch Specification 1.4.1 (ch. 6.8.4; pg. 54)

Topology
One control plane connection.

Results
Pass / Fail / Not applicable

Test recommendations
To validate the order counters may be use of interdependencies between messages in the bundle, while group features were utilized for the purpose of the test.

Additional remarks
The test may be not applicable as bundle is an optional feature and may not be supported by all OpenFlow switches.

NVCT.60.450 - Bundle messages - echo messages

Purpose
Verify if DUT supports exchanging echo messages during the creation and population of the bundle.

Methodology
Configure and connect DUT to controller. After control channel establishment, open bundle request using OFPBCT_OPEN_REQUEST. Use couple of OFPT_BUNDLE_ADD_MESSAGE messages to add different flows matching on a named field (under the Pre-requisites for the match). Close bundle request using OFPBCT_CLOSE_REQUEST and commit it using OFPBCT_COMMIT_REQUEST.
At different stages (after OFPT_BUNDLE_ADD_MESSAGE and OFPBCT_CLOSE_REQUEST messages) exchange echo messages with DUT, verify if switch replies properly.

Specification text
The switch must support exchanging echo request and echo reply messages during the creation and population of the bundle, the switch must reply to an echo request without waiting for the end of the bundle. Echo request and echo reply messages can not be included in a bundle. Similarly, asynchronous messages generated by the switch are not impacted by the bundle, the switch must send status events without waiting for the end of the bundle.
OpenFlow Switch Specification 1.4.1 (ch. 6.8.5; pg. 54)

Topology
One control plane connection.

Results
Pass / Fail / Not applicable

Additional remarks
The test may be not applicable as bundle is an optional feature and may not be supported by all OpenFlow switches.

NVCT.70 - OpenFlow Protocol v. 1.5 features

The OpenFlow Protocol v. 1.5 test suite verifies the DUT behaviour in the context of the most important features introduced in version 1.5 of OFP.

Remarks
The most valuable features introduced in version 1.5 of OpenFlow Protocol were selected. The aim is to verify if DUT supports those emerging features. Such a approach is reasonable as only few switches are fully compatible with OFP v1.5 while some of them reports to support selected features. Basic reference for all the tests is OpenFlow Switch Specification 1.5.1.

One of the most valuable features introduced in OpenFlow are egress tables which enable processing to be done in the context of the output port.
(OpenFlow Switch Specification 1.5.1, ch. B.19.1, pg. 277)

Version 1.5 introduces a flexible encoding, OpenFlow eXtensible Statistics (OXS), to encode any arbitrary flow entry statistics (EXT-334). Existing flow entry statistics are redefined as standard OXS fields.
(OpenFlow Switch Specification 1.5.1, ch. B.19.3, pg. 278)

Polling flow entry statistics can induce high overhead and utilisation for the switch. A new statistics trigger mechanism enable statistics to be automatically sent to the controller based on various statistics thresholds (EXT-335).
(OpenFlow Switch Specification 1.5.1, ch. B.19.4, pg. 279)

The existing Set-Field action allows to set a header field or pipeline field with a static value. The new Copy-Field action allows to copy the value from one header or pipeline field into another header or pipeline field (EXT-320).
(OpenFlow Switch Specification 1.5.1, ch. B.19.5, pg. 279)

A new OXM field OFPXMT_OFB_TCP_FLAGS has been added to match the flag bits in the TCP header (EXT-109). This field allows to match all flags, such as SYN, ACK and FIN, and may be used to detect the start and the end of TCP connections.
(OpenFlow Switch Specification 1.5.1, ch. B.19.7, pg. 279)

The previous version of the specification introduced bundle messages. A bundle is a sequence of OpenFlow modification requests from the controller that is applied as a single OpenFlow operation. The current specification extends the bundle feature (EXT-340), allowing: Scheduled bundles: a bundle commit message may include an execution time, specifying when the switch is expected to commit the bundle. Bundle features request: allows a controller to query a switch about its bundle capabilities, including whether it supports atomic bundles, ordered bundles, and scheduled bundles.
(OpenFlow Switch Specification 1.5.1, ch. B.19.11, pg. 280)

The new Copy-Field action allows to copy the value from one header or pipeline field into another header or pipeline field. New instruction OFPAT_COPY_FIELD to copy value between fields. Source and destination fields defined as OXM IDs, support standard OXMs, class based OXMs and experimenter OXMs. The Copy-Field instruction can copy bit sub-fields of any length from source to destination at any offset. New Table Features property to express support for Copy-Field.
(OpenFlow Switch Specification 1.5.1, ch. B.19.5, pg. 279)

NVCT.70.10 - Egress table - feature supported

NVCT.70.20 - Egress table - 0 table verification

NVCT.70.30 - Egress table - ingress table flag verification

NVCT.70.110 - First egress table - wrong table id

NVCT.70.120 - First egress table - setting flag

NVCT.70.130 - First egress table - clear flag

NVCT.70.140 - First egress table - bulk flag set

NVCT.70.210 - Ingress and egress pipeline - goto from ingress to ingress

NVCT.70.215 - Ingress and egress pipeline - invalid goto from ingress to egress

NVCT.70.220 - Ingress and egress pipeline - goto from egress to egress

NVCT.70.230 - Ingress and egress pipeline - invalid goto from egress to ingress

NVCT.70.240 - Ingress and egress pipeline - invalid action in egress table

NVCT.70.250 - Ingress and egress pipeline - egress table match

NVCT.70.260 - Ingress and egress pipeline - drop in output port context

NVCT.70.270 - Ingress and egress pipeline - modify copy in output port context

NVCT.70.280 - Ingress and egress pipeline - action set output in egress table

NVCT.70.290 - Ingress and egress pipeline - infinite loop

NVCT.70.310 - Extended flow statistics - OXS TLV format

NVCT.70.320 - Extended flow statistics - flow duration

NVCT.70.330 - Extended flow statistics - aggregated flows number

NVCT.70.340 - Extended flow statistics - packet count

NVCT.70.345 - Extended flow statistics - aggregated packet count

NVCT.70.350 - Extended flow statistics - byte count

NVCT.70.355 - Extended flow statistics - aggregated byte count

NVCT.70.360 - Extended flow statistics - idle time

NVCT.70.370 - Extended flow statistics - flow removed message

NVCT.70.410 - Flow statistics trigger - flow duration

NVCT.70.420 - Flow statistics trigger - idle time

NVCT.70.430 - Flow statistics trigger - packet count

NVCT.70.440 - Flow statistics trigger - byte count

NVCT.70.450 - Flow statistics trigger - only first

NVCT.70.460 - Periodic flow statistics trigger - flow duration

NVCT.70.470 - Periodic flow statistics trigger - idle time

NVCT.70.480 - Periodic flow statistics trigger - packet count

NVCT.70.490 - Periodic flow statistics trigger - byte count

NVCT.70.510 - OXM_OF_TCP_FLAGS: TCP Flag - ACK

Purpose
Verify that the switch is able to match on the OXM field OXM_OF_TCP_FLAGS as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) against ACK flag, action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered

Specification text
OXM_OF_TCP_SRC
Bits: 12
Mask: Yes
Prerequisite: IP PROTO=6
Description: TCP flags
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 81)

TCP flags matching A new OXM field OFPXMT_OFB_TCP_FLAGS has been added to match the flag bits in the TCP header (EXT-109). This field allows to match all flags, such as SYN, ACK and FIN, and may be used to detect the start and the end of TCP connections.
(OpenFlow Switch Specification 1.5.1, ch. B.19.7, pg. 279)

The field OXM_OF_TCP_FLAGS matches the flag bits in a TCP header. Considering bit 0 the least significant bit, it specifically contains:
• Bits 0–5: The original TCP flags defined in RFC 793
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 83)

Topology
One control plane connection and three data plane connections.

Results
Pass / Fail

NVCT.70.520 - OXM_OF_TCP_FLAGS: TCP Flag - PSH

Purpose
Verify that the switch is able to match on the OXM field OXM_OF_TCP_FLAGS as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) against PSH flag, action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered

Specification text
OXM_OF_TCP_SRC
Bits: 12
Mask: Yes
Prerequisite: IP PROTO=6
Description: TCP flags
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 81)

TCP flags matching A new OXM field OFPXMT_OFB_TCP_FLAGS has been added to match the flag bits in the TCP header (EXT-109). This field allows to match all flags, such as SYN, ACK and FIN, and may be used to detect the start and the end of TCP connections.
(OpenFlow Switch Specification 1.5.1, ch. B.19.7, pg. 279)

The field OXM_OF_TCP_FLAGS matches the flag bits in a TCP header. Considering bit 0 the least significant bit, it specifically contains:
• Bits 0–5: The original TCP flags defined in RFC 793
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 83)

Topology
One control plane connection and three data plane connections.

Results
Pass / Fail

NVCT.70.530 - OXM_OF_TCP_FLAGS: TCP Flag - RST

Purpose
Verify that the switch is able to match on the OXM field OXM_OF_TCP_FLAGS as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) against RSTflag, action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered

Specification text
OXM_OF_TCP_SRC
Bits: 12
Mask: Yes
Prerequisite: IP PROTO=6
Description: TCP flags
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 81)

TCP flags matching A new OXM field OFPXMT_OFB_TCP_FLAGS has been added to match the flag bits in the TCP header (EXT-109). This field allows to match all flags, such as SYN, ACK and FIN, and may be used to detect the start and the end of TCP connections.
(OpenFlow Switch Specification 1.5.1, ch. B.19.7, pg. 279)

The field OXM_OF_TCP_FLAGS matches the flag bits in a TCP header. Considering bit 0 the least significant bit, it specifically contains:
• Bits 0–5: The original TCP flags defined in RFC 793
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 83)

Topology
One control plane connection and three data plane connections.

Results
Pass / Fail

NVCT.70.540 - OXM_OF_TCP_FLAGS: TCP Flag - SYN

Purpose
Verify that the switch is able to match on the OXM field OXM_OF_TCP_FLAGS as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) against SYN flag, action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered

Specification text
OXM_OF_TCP_SRC
Bits: 12
Mask: Yes
Prerequisite: IP PROTO=6
Description: TCP flags
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 81)

TCP flags matching A new OXM field OFPXMT_OFB_TCP_FLAGS has been added to match the flag bits in the TCP header (EXT-109). This field allows to match all flags, such as SYN, ACK and FIN, and may be used to detect the start and the end of TCP connections.
(OpenFlow Switch Specification 1.5.1, ch. B.19.7, pg. 279)

The field OXM_OF_TCP_FLAGS matches the flag bits in a TCP header. Considering bit 0 the least significant bit, it specifically contains:
• Bits 0–5: The original TCP flags defined in RFC 793
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 83)

Topology
One control plane connection and three data plane connections.

Results
Pass / Fail

NVCT.70.550 - OXM_OF_TCP_FLAGS: TCP Flag - FIN

Purpose
Verify that the switch is able to match on the OXM field OXM_OF_TCP_FLAGS as a single header field match.

Methodology
Configure and connect DUT to controller. After control channel establishment, add a flow matching on the named field (under the given Pre-requisites for the match) against FIN flag, action is forwarding to an output port. Send a matching packet on the data plane. Verify that the packet is received only at the port specified in the flow action. Send a non-matching packet, verify that the flow does not forward it, and that a table-miss is triggered

Specification text
OXM_OF_TCP_SRC
Bits: 12
Mask: Yes
Prerequisite: IP PROTO=6
Description: TCP flags
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 81)

TCP flags matching A new OXM field OFPXMT_OFB_TCP_FLAGS has been added to match the flag bits in the TCP header (EXT-109). This field allows to match all flags, such as SYN, ACK and FIN, and may be used to detect the start and the end of TCP connections.
(OpenFlow Switch Specification 1.5.1, ch. B.19.7, pg. 279)

The field OXM_OF_TCP_FLAGS matches the flag bits in a TCP header. Considering bit 0 the least significant bit, it specifically contains:
• Bits 0–5: The original TCP flags defined in RFC 793
(OpenFlow Switch Specification 1.5.1, ch. 7.2.3.8, pg. 83)

Topology
One control plane connection and three data plane connections.

Results
Pass / Fail

NVCT.70.610 - Scheduled bundle - correct usage

NVCT.70.620 - Scheduled bundle - discarding

NVCT.70.630 - Scheduled bundle - past time

NVCT.70.640 - Scheduled bundle - past tolerance exceeded

NVCT.70.650 - Scheduled bundle - future tolerance exceeded

NVCT.70.710 - Copy - from header field to a packet register pipeline

Purpose
Verify if copy-field action may copy data from header file to a packet register pipeline

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on the named field (under the given Pre-requisites for the match), with an output action to egress port and a supported copy-field action from header field to the selected packet register pipeline. Send a matching packet on the data plane via a certain ingress port. Verify that selected header field value was copied to the packet register pipeline.

Specification text
Copy-Field src field type dst field type. The Copy-Field action may copy data between any header or pipeline fields. It is typically used to copy data from a header field to a packet register pipeline field or from a packet register pipeline field to a header field, and in some cases from a header field to another header field. A switch may not support all combinations of copies between header or pipeline fields.
OpenFlow Switch Specification 1.5.1 (ch. 5.8; pg. 29)

The effect of copy-field actions in the action set is undefined due to race conditions, and therefore its implementation discouraged.
OpenFlow Switch Specification 1.5.1 (ch. 5.6; pg. 26)

Topology
One control plane and two data plane connections.

Results
Pass / Fail / Not applicable

Additional remarks
If switch do not support particular copy action then test is not applicable.

NVCT.70.720 - Copy - from packet register pipeline to header field

Purpose
Verify if copy-field action may copy data from packet register pipeline to header file

Methodology
Configure and connect DUT to controller. After control channel establishment, set-value to packet register pipeline. Create an ofp_flow_mod matching on the named field (under the given Pre-requisites for the match), with an output action to egress port and a supported copy-field action from the selected packet register pipeline to header field. Send a matching packet on the data plane via a certain ingress port. On the egress port verify that selected register value was copied to the header field.

Specification text
Copy-Field src field type dst field type. The Copy-Field action may copy data between any header or pipeline fields. It is typically used to copy data from a header field to a packet register pipeline field or from a packet register pipeline field to a header field, and in some cases from a header field to another header field. A switch may not support all combinations of copies between header or pipeline fields.
OpenFlow Switch Specification 1.5.1 (ch. 5.8; pg. 29)

The effect of copy-field actions in the action set is undefined due to race conditions, and therefore its implementation discouraged.
OpenFlow Switch Specification 1.5.1 (ch. 5.6; pg. 26)

Topology
One control plane and two data plane connections.

Results
Pass / Fail / Not applicable

Additional remarks
If switch do not support particular copy action then test is not applicable.

NVCT.70.730 - Copy - from one header field to another header field

Purpose
Verify if copy-field action may copy data from one header field to another header file

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on the named field (under the given Pre-requisites for the match), with an output action to egress port and a supported copy-field action from one header field to another header field. Send a matching packet on the data plane via a certain ingress port. Verify that selected header field value was copied to another header field.

Specification text
Copy-Field src field type dst field type. The Copy-Field action may copy data between any header or pipeline fields. It is typically used to copy data from a header field to a packet register pipeline field or from a packet register pipeline field to a header field, and in some cases from a header field to another header field. A switch may not support all combinations of copies between header or pipeline fields.
OpenFlow Switch Specification 1.5.1 (ch. 5.8; pg. 29)

The effect of copy-field actions in the action set is undefined due to race conditions, and therefore its implementation discouraged.
OpenFlow Switch Specification 1.5.1 (ch. 5.6; pg. 26)

Topology
One control plane and two data plane connections.

Results
Pass / Fail / Not applicable

Additional remarks
It is recommended to copy eth_src to eth_dst
If switch do not support particular copy action then test is not applicable.

NVCT.70.740 - Copy - invalid offsets

Purpose
Verify if DUT rejects copy field actions with invalid offsets.

Methodology
Configure and connect DUT to controller. After control channel establishment, create an ofp_flow_mod matching on the named field (under the given Pre-requisites for the match), with an output action to egress port and a supported copy-field action with invalid offset. Validate if an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_SET_ARGUMENT code was returned.

Specification text
The switch must reject Copy Field actions for which src_offset+n_bits is greater than the width of src_oxm_id or dst_offset+n_bits is greater than the width of dst_oxm_id with a Bad Set Argument error message (see 7.5.4.3).
OpenFlow Switch Specification 1.5.1 (ch. 7.2.6.8; pg. 100)

If a Copy-Field action specifies invalid offsets, the switch must return an ofp_error_msg with OFPET_BAD_ACTION type and OFPBAC_BAD_SET_ARGUMENT code.
OpenFlow Switch Specification 1.5.1 (ch. 7.5.4.3; pg. 188)

Topology
One control plane connection.

Results
Pass / Fail / Not applicable

Test recommendations
Test both options of invalid offsets: regarding to the src_oxm_id and dst_oxm_id.

Additional remarks
If switch do not support particular copy action then test is not applicable.

6.2. Technical Specification

Each virtual machine composing NVCT system may run on the commodity server and any specialized hardware is needed. Only required number of network interfaces is needed to perform tests according to the specification. Also virtualization software can be utilized to run all the system logical components in the single physical machine. Full virtualization allows for remote system deployment. Therefore, all technical specification of the underlying hardware depends on the specification of the hardware actually used.

Most tests results with simple pass/fail flag indicating if DUT is compliant with respect to the tested functionality. However, also Not Applicable (N/A) result is possible to indicate that for some reasons tests could not be run. Possible causes are: DUT notified lack of support for the functionality being tested or needed to perform the test. L2_FAIL fail flag reason indicates that problems in layer two causes test failure.

However, also some additional parameters are measured by the system. Most of them are performance parameters, all of them are explained in specific tests. Below summary information about those parameters is provided:

- Numerical value indicating capacity of each table - denoting number of entries that can be added to the specific table
- Numerical value of the latency - time period between start and end of the specific action, for example between sending packet by the generator and receiving it by the analyzer
- Numerical value of the latency in a function of total number of flows installed in the table - denoting dependency of latency measured for the specific action from the occupancy level of the selected table
- Numerical value of the flow add rate - indicating how fast new flows can be added to the table
- Numerical value of packet out per second - indicating switch performance regarding how fast packet out messages can be handled
- Memory size in bytes - buffer capacity
- Table miss performance indicator - informs if OpenFlow table miss action performed in DIT is better/equal/worse (in terms of latency) than default switch behaviour
- Throughput value - is the amount of data transmitted in the time unit
- Indicator if flow statistics tracking has/has not impact on flow handling performance.- informs if statistics gathering influence normal switch behaviour like switching data, adding entries to the table, etc. depending on the test suite

In order to ensure time synchronization between traffic generator and analyzer dedicated Network Time Protocol (NTP) server was deployed and adjusted to the system needs.

6.3. Licenses

NVCT system utilizes several 3rd party modules and libraries distributed under different licenses. Below we present complete list those modules along with license information:

- Netifaces - under MIT license
- Ntpserver - under MIT license
- ntplib - under LGPL license
- PyInstaller - under GPL license
- Ryu - under Apache 2.0 license
- Scapy - under GPLv2 license
- Zeroconf - under LGPL license, the library source code was slightly modified to improve exceptions handling, diffs are attached to the NVCT distribution as well as provided below under LGPL license

---

```
 zeroconf/zeroconf.py | 116 ++++++++++++++++++++++++++++++++----------------------
 1 file changed, 64 insertions(+), 52 deletions(-)

diff --git a/zeroconf/zeroconf.py b/zeroconf/zeroconf.py
index 5fad710..960edc1 100644
--- a/zeroconf/zeroconf.py
+++ b/zeroconf/zeroconf.py
@@ -1098,28 +1098,32 @@ class Engine(threading.Thread):
         self.start()

     def run(self):
-        while not self.zc.done:
-            with self.condition:
-                rs = self.readers.keys()
-                if len(rs) == 0:
-                    # No sockets to manage, but we wait for the timeout
-                    # or addition of a socket
-                    self.condition.wait(self.timeout)
-
-            if len(rs) != 0:
-                try:
-                    rr, wr, er = select.select(rs, [], [], self.timeout)
-                    if not self.zc.done:
-                        for socket_ in rr:
-                            reader = self.readers.get(socket_)
-                            if reader:
-                                reader.handle_read(socket_)
-
-                except socket.error as e:
-                    # If the socket was closed by another thread, during
-                    # shutdown, ignore it and exit
-                    if e.errno != socket.EBADF or not self.zc.done:
-                        raise
+        try:
+            while not self.zc.done:
+                with self.condition:
+                    rs = self.readers.keys()
+                    if len(rs) == 0:
+                        # No sockets to manage, but we wait for the timeout
+                        # or addition of a socket
+                        self.condition.wait(self.timeout)
+
+                if len(rs) != 0:
+                    try:
+                        rr, wr, er = select.select(rs, [], [], self.timeout)
+                        if not self.zc.done:
+                            for socket_ in rr:
+                                reader = self.readers.get(socket_)
+                                if reader:
+                                    reader.handle_read(socket_)
+
+                    except socket.error as e:
+                        # If the socket was closed by another thread, during
+                        # shutdown, ignore it and exit
+                        if e.errno != socket.EBADF or not self.zc.done:
+                            raise
+        except:
+            # silently ignore all exceptions
+            pass

     def add_reader(self, reader, socket_):
         with self.condition:
```

```
@@ -1186,15 +1190,19 @@ class Reaper(threading.Thread):
         self.start()

     def run(self):
-        while True:
-            self.zc.wait(10 * 1000)
-            if self.zc.done:
-                return
-            now = current_time_millis()
-            for record in self.zc.cache.entries():
-                if record.is_expired(now):
-                    self.zc.update_record(now, record)
-                    self.zc.cache.remove(record)
+        try:
+            while True:
+                self.zc.wait(10 * 1000)
+                if self.zc.done:
+                    return
+                now = current_time_millis()
+                for record in self.zc.cache.entries():
+                    if record.is_expired(now):
+                        self.zc.update_record(now, record)
+                        self.zc.cache.remove(record)
+        except:
+            # silently ignore all exceptions
+            pass


 class Signal(object):
@@ -1320,27 +1328,31 @@ class ServiceBrowser(threading.Thread):
     def run(self):
         self.zc.add_listener(self, DNSQuestion(self.type, _TYPE_PTR, _CLASS_IN))

-        while True:
-            now = current_time_millis()
-            if len(self._handlers_to_call) == 0 and self.next_time > now:
-                self.zc.wait(self.next_time - now)
-            if self.zc.done or self.done:
-                return
-            now = current_time_millis()
-            if self.next_time <= now:
-                out = DNSOutgoing(_FLAGS_QR_QUERY)
-                out.add_question(DNSQuestion(self.type, _TYPE_PTR, _CLASS_IN))
-                for record in self.services.values():
-                    if not record.is_expired(now):
-                        out.add_answer_at_time(record, now)
-
-                self.zc.send(out)
-                self.next_time = now + self.delay
-                self.delay = min(20 * 1000, self.delay * 2)
-
-            if len(self._handlers_to_call) > 0 and not self.zc.done:
-                handler = self._handlers_to_call.pop(0)
-                handler(self.zc)
+        try:
+            while True:
+                now = current_time_millis()
+                if len(self._handlers_to_call) == 0 and self.next_time > now:
+                    self.zc.wait(self.next_time - now)
+                if self.zc.done or self.done:
+                    return
+                now = current_time_millis()
```

```
+                if self.next_time <= now:
+                    out = DNSOutgoing(_FLAGS_QR_QUERY)
+                    out.add_question(DNSQuestion(self.type, _TYPE_PTR, _CLASS_IN))
+                    for record in self.services.values():
+                        if not record.is_expired(now):
+                            out.add_answer_at_time(record, now)
+
+                    self.zc.send(out)
+                    self.next_time = now + self.delay
+                    self.delay = min(20 * 1000, self.delay * 2)
+
+                if len(self._handlers_to_call) > 0 and not self.zc.done:
+                    handler = self._handlers_to_call.pop(0)
+                    handler(self.zc)
+        except:
+            # ignore all exceptions silently
+            pass


 class ServiceInfo(object):
--
```

Glossary

Command Line Interface (CLI) - the user interface to communicate with NVCT system, UNIX-based textual command line is utilized in current version of the system so users familiar with unix shell can easily use the NVCT system

configuration file - file internal for the NVCT system in which all the configuration parameters can be configured according to the Configuration Section. The file is called dut_config.cfg

configuration parameter - data provided by the user in order to perform tests, some parameters are optional, some parameters are obligatory to perform selected tests or even whole test groups, all configuration parameters are provided in configuration file, for details please refer to the Configuration Section

***controller* -** please see *nvct-controller*

datapath id (dpid) - OpenFlow datapath identifier of DUT. Each OpenFlow instance on a switch is identified by a Datapath Identifier. This is a 64 bit number determined as follows according to the OpenFlow specification. The lower 48 bits are intended for the switch MAC address, while the top 16 bits are up to the implementer. An example use of the top 16 bits would be a VLAN ID to distinguish multiple virtual switch instances on a single physical switch

***dataplane* -** physical connections between *nvct-dataplane* and DUT

Device Under Test (DUT) - Device Under Test (DUT) is a switch pretending to be compliant with the OpenFlow protocol (OpenFlow Switch Specification). The aim of the NVCT system is to test if DUT is really compatible

DUT manual - user guide of device being tested, provided by the DUT vendor

Flow - stream of packets identified by values in selected header files

L2-switch / legacy switch - networking equipment operating on 2nd level in OSI/ISO architecture, L2-switch don't have to be compliant with OpenFlow Protocol, however many OpenFlow enabled switches are able to operate in legacy mode

NVCT - NetVizion Compliance Tester, this product, which main capabilities and features are described in this user guide

***nvct-controller* -** SDN controller being component of the NVCT system architecture. It is Ryu open source controller developed in Python modified for the system and integration purposes. The main responsibility of Ryu is to generate OpenFlow messages to the DUT. This customized SDN controller is also referred as *controller*.

***nvct-dataplane* -** both data plane components of the NVCT system: traffic generator and analyzer. Along with physical connections with DUT creates a *dataplane*

***nvct-tester* -** the module being brain of NVCT system that integrates the whole process of testing. It is integrated with module responsible for analysis of control channel between SDN controller and DUT. Both the main component as well as analyzer are written in Python programming language. Also referred as *tester*.

OpenFlow Protocol - allows for communication between separated data plane and control plane

OpenFlow enabled port - Ethernet port which may be either physical or logical port which configuration may be modified by an OpenFlow controller; however, its state is an indication of current link status and is read-only. Each port has an associated Port ID, which may not come from the reserved range of Port IDs. Additionally, a port has an Ethernet MAC address and human readable description.

Open Network Foundation (ONF) - Open Networking Foundation (ONF) is a user-driven organization dedicated to the promotion and adoption of Software-Defined Networking (SDN) through open standards development.

Public Key Infrastructure - correlates cryptographic keys with identities managing public-key encryption. Needed for the purpose of secure communication between *nvct-controller* and DUT

Ryu - written in Python programming language SDN controller that allows for OpenFlow based communication with OpenFlow enabled switches. Modified version of Ryu is a base for the *nvct-controller* component

Software Defined Network (SDN) - the concept of separation between data plane and control plane for the purpose of more elastic and automatic network configuration

system component / system module - modules if the NVCT system: *nvct-tester*, *nvct-controller* or *nvct-dataplane*

SDN-switch - switch compatible with OpenFlow Protocol, please note that OpenFlow is one of the possible implementations of SDN concept, however NVCT system is focused on OpenFlow Protocol

test - the smallest unit of NVCT system operation, the result of the test inform if tested functionality is supported by the DUT, not-supported or test was skipped due to the provided reason

test case - tests are grouped in cases to ease operation, administration and configuration of the system

test operator - NVCT system user

test specification - refers to one of two documents, ONF Conformance Test Specification for OpenFlow Switch Specification ver. 1.3.4 or tests designed and implemented for the purpose of NVCT system which specification was attached to this user guide in Specifications Section.

***tester* -** please see *nvct-tester*

user - a person that uses NVCT system, which means that he/she installs the system, configures it, performs test, and gather results

Virtual Machine (VM) - software computer that, like a physical computer, runs an operating system and applications. The virtual machine is comprised of a set of specification and configuration files and is backed by the physical resources of a host. In the default architecture of NVCT, each component (*nvct-controller, nvct-dataplane, and nvct-tester*) are an applications running on separate virtual machines.