Test Vector Framework for Stratum Enabled Switches

Abhilash Endurthi, You Wang
Open Networking Foundation
Outline

- Introduction
- Test Vector Details
- Test Vector Runner Details
- Next Steps
Introduction

- What are we trying to achieve?
  - Develop set of vendor agnostic tests to certify a switch as Stratum compliant
  - Develop a framework (runner) to execute the tests
- How?
  - Using black box methodology
  - Data driven tests
- What is our device under test?
  - Switches running Stratum
  - Switches that comply with Stratum open APIs (gNMI, gNOI, P4Runtime)

Source: Black Box Testing of Stratum Enabled Switches, ONF Connect 2018
Test Vectors Overview

• Separate test definitions from test infra
  ○ Vendors use different infra/frameworks/programming languages for testing
  ○ A way to define tests so that they could be easily supported by various test infra

• A compact way of defining test input/output
  ○ \(TV = \{TC_i\}\) where \(TC_i = (Actions_i, Expectations_i)\)
  ○ Actions and Expectations: Open APIs accesses and external stimuli (port events, dataplane packet IO, etc.)

Source: Black Box Testing of Stratum Enabled Switches, ONF Connect 2018
Black Box Testing with Test Vectors

Source: Black Box Testing of Stratum Enabled Switches, ONF Connect 2018
Outline

● Introduction
● Test Vector Details
● Test Vector Runner Details
● Next Steps
Test Vector

- Coded using protobufs
- TV protobuf definition is open sourced with Stratum
- gNMI, gNOI and P4Runtime also use protobufs
- Language specific source code can be generated for classes using protoc compiler
Test Vector Definition

- **Target**
  - test_case_id

- **Test Case**
  - action_group_id
    - Parallel Action Group
    - Sequential Action Group
    - Randomized Action Group

- **Test Vector**
  - Expectation

Optional attribute

Oneof attribute

Repeated attribute
Expectation Definition

- **expectation_id**
  - **Config Expectation**
    - gnmi.GetRequest
    - gnmi.GetResponse
  - **Control Plane Expectation**
    - Read Expectation
    - Packet In Expectation
    - Pipeline Config Expectation
  - **Telemetry Expectation**
    - Requirement
    - Action Group
    - gnmi.SubscribeRequest
    - gnmi.SubscribeResponse
  - **Data Plane Expectation**
    - Traffic Expectation

Optional attribute
Oneof attribute
Repeated attribute
Test Vector Example

install�

- action_groups: <
  - action_group_id: “ag1”
- action_groups: <
  - action_group_id: “ag2”
- expectations: <
  - expectation_id: “e1”
- expectations: <
  - expectation_id: “e2”
  test_case_id: “tc1”
  test_cases: <
  - test_case_id: “tc2”

control_plane_operation: <
  write_operation: <
  - p4_write_request: <
  - p4_read_request: <
  - p4_read_response: <

data_plane_operation: <
  traffic_stimulus: <
  - packets: <
    - payload: “”
  ports: 2

data_plane_expectation: <
  traffic_expectation: <
  - packets: <
    - payload: “”
  ports: 2
test_cases:
  action_groups:
    action_group_id: "ag1"
    action_groups:
      action_group_id: "ag2"
    expectations:
      expectation_id: "e1"
    expectations:
      expectation_id: "e2"
  test_case_id: "tc1"
  test_cases:
    test_case_id: "tc2"

sequential_action_group:
  actions:
    control_plane_operation:
      write_operation:
        p4_write_request:
          ...
    data_plane_operation:
      traffic_stimulus:
        ...
    ...
    >
    >
    >
    >
    actions:
      data_plane_operation:
        traffic_stimulus:
          ...
    >
    >
    >
    >
**Test Vector Example**

```xml
<test_cases>
  <action_groups>
    <...>
    <action_group_id>“ag1”</action_group_id>
    <...>
  </action_groups>
  <expectations>
    <...>
    <expectation_id>“e1”</expectation_id>
    <...>
  </expectations>
  <test_case_id>“tc1”</test_case_id>
  <...>
  <test_case_id>“tc2”</test_case_id>
  <...>
</test_cases>

<control_plane_expectation>
  <read_expectation>
    <p4_read_request>
      <...>
    </p4_read_request>
    <p4_read_response>
      <...>
    </p4_read_response>
  </read_expectation>
  <...>
</control_plane_expectation>

<data_plane_expectation>
  <traffic_expectation>
    <packets>
      <payload>“”</payload>
    </packets>
    <ports>2</ports>
  </traffic_expectation>
  <...>
</data_plane_expectation>
```
Test Vector Example
Test Vectors Implemented

- **p4runtime**
  - PktIoOutDirectToDataPlaneTest
  - PktIoOutToIngressPipelineAclPuntToCpuTest
  - PktIoOutToIngressPipelineAclRedirectToPortTest
  - PktIoOutToIngressPipelineL3ForwardingTest
  - PacketIoOutDirectLoopbackPortAclTest
  - PacketIoOutDirectLoopbackL3ForwardingTest
  - RedirectDataplaneToCpuACLTest
  - RedirectDataplaneToCpuNextHopTest
  - RedirectDataplaneToDataplaneTest
  - L3ForwardTest

- **gnmi**
  - Subscribe_HEALTH_Indicator
  - Config_expectation_1
  - Config_expectation_2
  - ...
  - Config_expectation_36

- **e2e**
  - SubRedirectDataplaneToDataplane

- **Targets supported:** bmv2, Barefoot Tofino, Broadcom Tomahawk
Outline

- Introduction
- Test Vector Details
- **Test Vector Runner Details**
- Next Steps
Test Vector Runner

- Reference implementation written in Golang
  - Uses Go testing framework
- Target independent
  - Runs with bmv2/hardware switches
  - By reading different input files: target/port-map/test vectors
- Easy to deploy
  - Provides tools to deploy and run as container/binary
Test Vectors and Go Testing

Test Suite

Test Vector A

Test Case 1
Action 1: p4_write
Action 2: packet_out
... Expectation 1: gnmi_get
Expectation 2: packet_exp
...

Test Case 2
...

Test Vector B
...

```go
testing.main( []testing.InternalTest{
    testing.InternalTest{
        Name: "Test_Vector_A",
        F: func(t *testing.T) {
            t.Run("Test_Case_1", func(t *testing.T) {
                ProcessP4WriteRequest(request)
                ProcessPacketOutOperation(request)
            }...
            ProcessGnmiGetRequest(request)
            ProcessTrafficExpectation(packet, port)
        }...
        t.Run("Test_Case_2", func(t *testing.T) {})
    }...
    testing.InternalTest{
        Name: "Test_Vector_B",
        F: func(t *testing.T) {}}

    ...
})
```
TV Runner - Actions

- Requirement
  - Config Operation
    - gnmi.SetRequest
    - gnmi.SetResponse
  - Control Plane Operation
    - Write Operation
    - Packet Out Operation
    - Pipeline Config Operation
  - Management Operation
    - Security Operation
    - Diag Operation
    - System Operation
    - File Operation
  - Data Plane Stimulus
    - Traffic Stimulus
      - gnmi.Path
      - State
    - Port Stimulus
      - gnmi.Path
  - Alarm Stimulus

Implemented
- Not implemented

Optional attribute
- Oneof attribute
- Repeated attribute
Test Execution

```bashoot@1688040ffca3:-/tv_runner# make p4runtime
./tv_runner -test.v -tvDir=$HOME/tv/bmv2/p4runtime/ -tgFile=$HOME/tv/bmv2/target.pb.txt -portMapFile=tools/bmv2/port-map.json -logLevel=fatal

--- RUN L3ForwardTest
--- RUN L3ForwardTest/insert_write
--- RUN L3ForwardTest/packet_len_78_ttl_64_port_1
--- RUN L3ForwardTest/packet_len_78_ttl_64_port_2
--- RUN L3ForwardTest/delete_write
--- PASS: L3ForwardTest (9.29s)
  --- PASS: L3ForwardTest/insert_write (0.26s)
  --- PASS: L3ForwardTest/packet_len_78_ttl_64_port_1 (4.38s)
  --- PASS: L3ForwardTest/packet_len_78_ttl_64_port_2 (4.42s)
  --- PASS: L3ForwardTest/delete_write (0.23s)
--- RUN PktIoOutDirectToDataPlaneTest
--- RUN PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_64_port_1
--- RUN PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_64_port_2
--- RUN PktIoOutDirectToDataPlaneTest/packetout_len_1500_ttl_64_port_1
--- RUN PktIoOutDirectToDataPlaneTest/packetout_len_1500_ttl_64_port_2
--- RUN PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_0_port_1
--- RUN PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_0_port_2
--- PASS: PktIoOutDirectToDataPlaneTest (13.48s)
  --- PASS: PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_64_port_1 (2.23s)
  --- PASS: PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_64_port_2 (2.28s)
  --- PASS: PktIoOutDirectToDataPlaneTest/packetout_len_1500_ttl_64_port_1 (2.27s)
  --- PASS: PktIoOutDirectToDataPlaneTest/packetout_len_1500_ttl_64_port_2 (2.26s)
  --- PASS: PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_0_port_1 (2.26s)
  --- PASS: PktIoOutDirectToDataPlaneTest/packetout_len_78_ttl_0_port_2 (2.26s)
--- RUN RedirectDataplaneToDataplaneTest
--- RUN RedirectDataplaneToDataplaneTest/insert_write
--- RUN RedirectDataplaneToDataplaneTest/packet_len_78_ttl_64_port_1
--- RUN RedirectDataplaneToDataplaneTest/packet_len_78_ttl_64_port_1#01
--- RUN RedirectDataplaneToDataplaneTest/delete_write
--- PASS: RedirectDataplaneToDataplaneTest (9.35s)
  --- PASS: RedirectDataplaneToDataplaneTest/insert_write (0.19s)
  --- PASS: RedirectDataplaneToDataplaneTest/packet_len_78_ttl_64_port_1 (4.46s)
  --- PASS: RedirectDataplaneToDataplaneTest/packet_len_78_ttl_64_port_1#01 (4.50s)
  --- PASS: RedirectDataplaneToDataplaneTest/delete_write (0.20s)
```
Test Execution

```
root@1580a40ffca3:/tv_runner# make e2e LOG_LEVEL=info
./tv_runner -test.v -tvDir=/HOME/tv/bmv2/e2e/ -tgFile=HOME/tv/bmv2/target_pb.txt -portMapFile=tools/bmv2/port-map.json -logLevel=info
INFO[2019-09-09T20:52:34.424Z] Target: address:"localhost:50001" target_id:"t1"
INFO[2019-09-09T20:52:34.429Z] Setting up test suite...
--- RUN SubRedirectDataplaneToDataplane
INFO[2019-09-09T20:52:34.431Z] Setting up test...
--- RUN SubRedirectDataplaneToDataplane/write
INFO[2019-09-09T20:52:34.645Z] Test Case ID: insert_write
INFO[2019-09-09T20:52:34.645Z] Action Group ID: ag1
INFO[2019-09-09T20:52:34.647Z] Sending P4 write request
INFO[2019-09-09T20:52:34.649Z] Sending P4 write request
INFO[2019-09-09T20:52:34.652Z] Sending P4 write request
INFO[2019-09-09T20:52:34.651Z] ........................................................................................................
--- RUN SubRedirectDataplaneToDataplane/subscribe
INFO[2019-09-09T20:52:34.825Z] Test Case ID: subscribe
INFO[2019-09-09T20:52:34.825Z] Expectation ID: e1
INFO[2019-09-09T20:52:34.828Z] Sending subscription request
INFO[2019-09-09T20:52:34.831Z] Subscription responses are equal
INFO[2019-09-09T20:52:36.826Z] Sending packets to interface veth0
INFO[2019-09-09T20:52:36.925Z] Sending packet to interface veth0
INFO[2019-09-09T20:52:37.829Z] Subscription responses are equal
INFO[2019-09-09T20:52:37.832Z] Checking packets on interface veth2
ERROR[2019-09-09T20:52:38.332Z] Payloads of packet #1 don't match
INFO[2019-09-09T20:52:38.332Z] ........................................................................................................
--- RUN SubRedirectDataplaneToDataplane/delete_write
INFO[2019-09-09T20:52:38.585Z] Sending P4 write request
INFO[2019-09-09T20:52:38.587Z] Sending P4 write request
INFO[2019-09-09T20:52:38.596Z] Sending P4 write request
INFO[2019-09-09T20:52:38.597Z] ........................................................................................................
INFO[2019-09-09T20:52:38.597Z] Tearing down test...
--- FAIL: SubRedirectDataplaneToDataplane (4.17s)
    --- PASS: SubRedirectDataplaneToDataplane/insert_write (0.22s)
    --- FAIL: SubRedirectDataplaneToDataplane/subscribe (3.68s)
    --- PASS: SubRedirectDataplaneToDataplane/delete_write (0.27s)
FAIL
```
Deployment Scenarios

Scenario 1

Container

TV Runner

bmv2
Software Switch

gRPC
veth

Scenario 2

Hardware Switch

Server

TV Runner

gRPC

Data plane interfaces

TV Runner Repo

deploy

TV Repo

deploy

TV Runner Repo

deploy

TV Repo

deploy
Outline

- Introduction
- Test Vector Details
- Test Vector Runner Details
- Next Steps
Test Vector Generation - Current Approach

- Hand written test vectors
  - Tedious
  - Time consuming
  - Error prone
  - Hard to debug

- Semi automatically generated
  - P4RT write requests from stratum log
  - Pipeline config from P4RT generated binaries and json files
  - gNMI get operations using list of paths

- p4runtime
  - PktioOutDirectToDataPlaneTest
  - PktioOutToIngressPipelineAclPuntToCpuTest
  - PktioOutToIngressPipelineAclRedirectToPortTest
  - PktioOutToIngressPipelineL3ForwardingTest
  - PacketioOutDirectLoopbackPortAclTest
  - PacketioOutDirectLoopbackL3ForwardingTest
  - RedirectDataplaneToCpuACLTest
  - RedirectDataplaneToCpuNextHopTest
  - RedirectDataplaneToDataplaneTest
  - L3ForwardTest

- gNMI
  - Subscribe_Hand_Indicator
  - Config_expectation_1
  - Config_expectation_2
  - ...
  - Config_expectation_36
Test Vector Generation - Next Steps

- Automatic generation of test vectors based on input from
  - Chassis config
  - SDN controller trace
  - ATPG (Automatic Test Packet Generation)
More Testing Scenarios

Scenario 1
- Container
- Hardware Switch

Scenario 2
- Server

Scenario 3
- Hardware Switch
- TV Runner
- ASIC
- Port 1
- Port 2
- PktOut
- PktIn
- gRPC

Scenario 4
- Server
- TV Runner
- Traffic Generator
- gRPC

TV Runner Repo
- deploy

TV Repo
- deploy

Scenario 1
Scenario 2
Scenario 3
Scenario 4
Call for Community Help

- **Test Vectors**
  - Adding more test vectors to the repo
  - Adding test vector generators, utility functions for automated test vector generation

- **Test Vector Runner**
  - Support missing operations
  - Support more deployment scenarios
Thank You

Follow Up Links:
https://stratumproject.slack.com/
abhilash@opennetworking.org  you@opennetworking.org