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

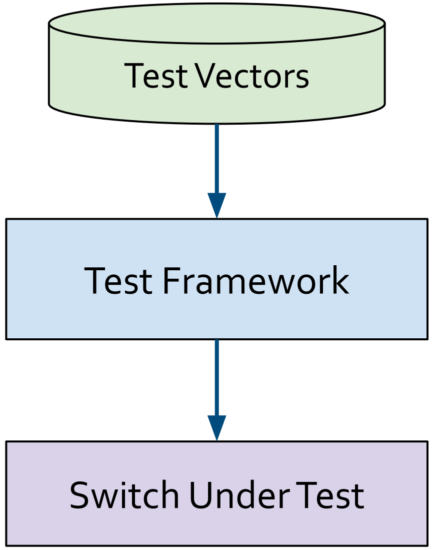
Introduction

- 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

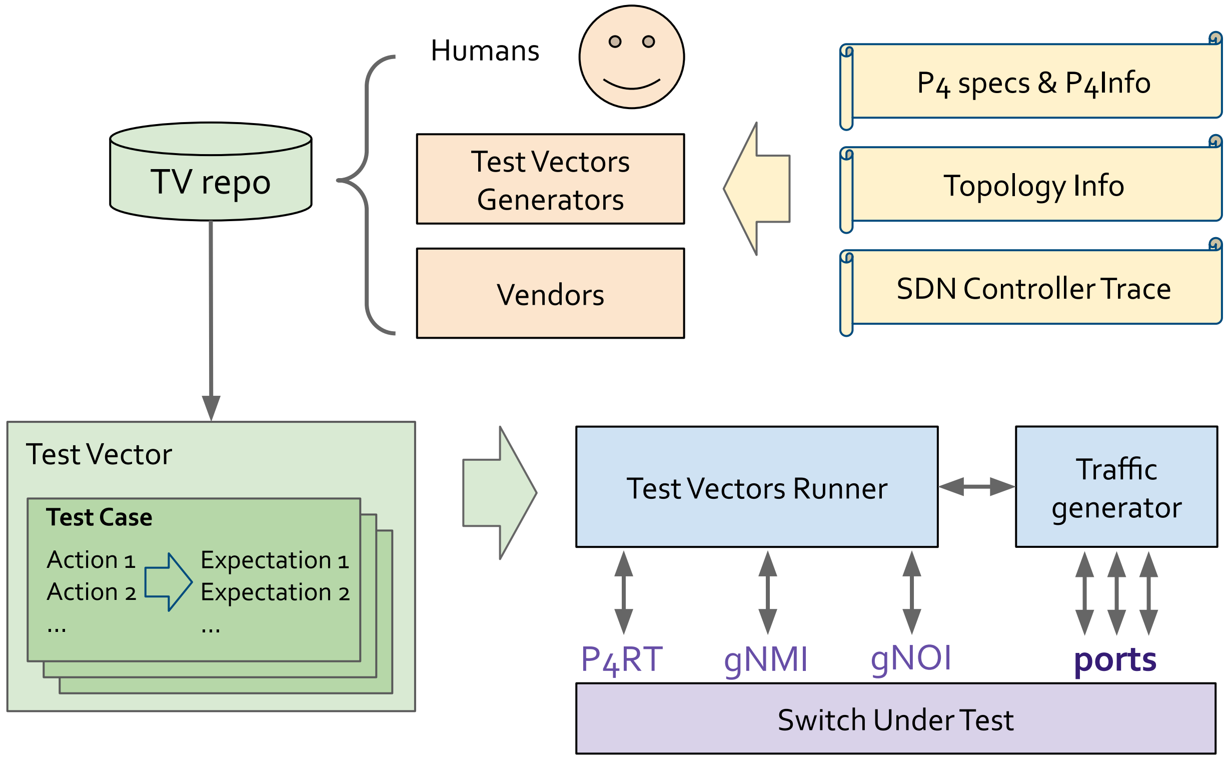
Test Vectors Overview

- 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

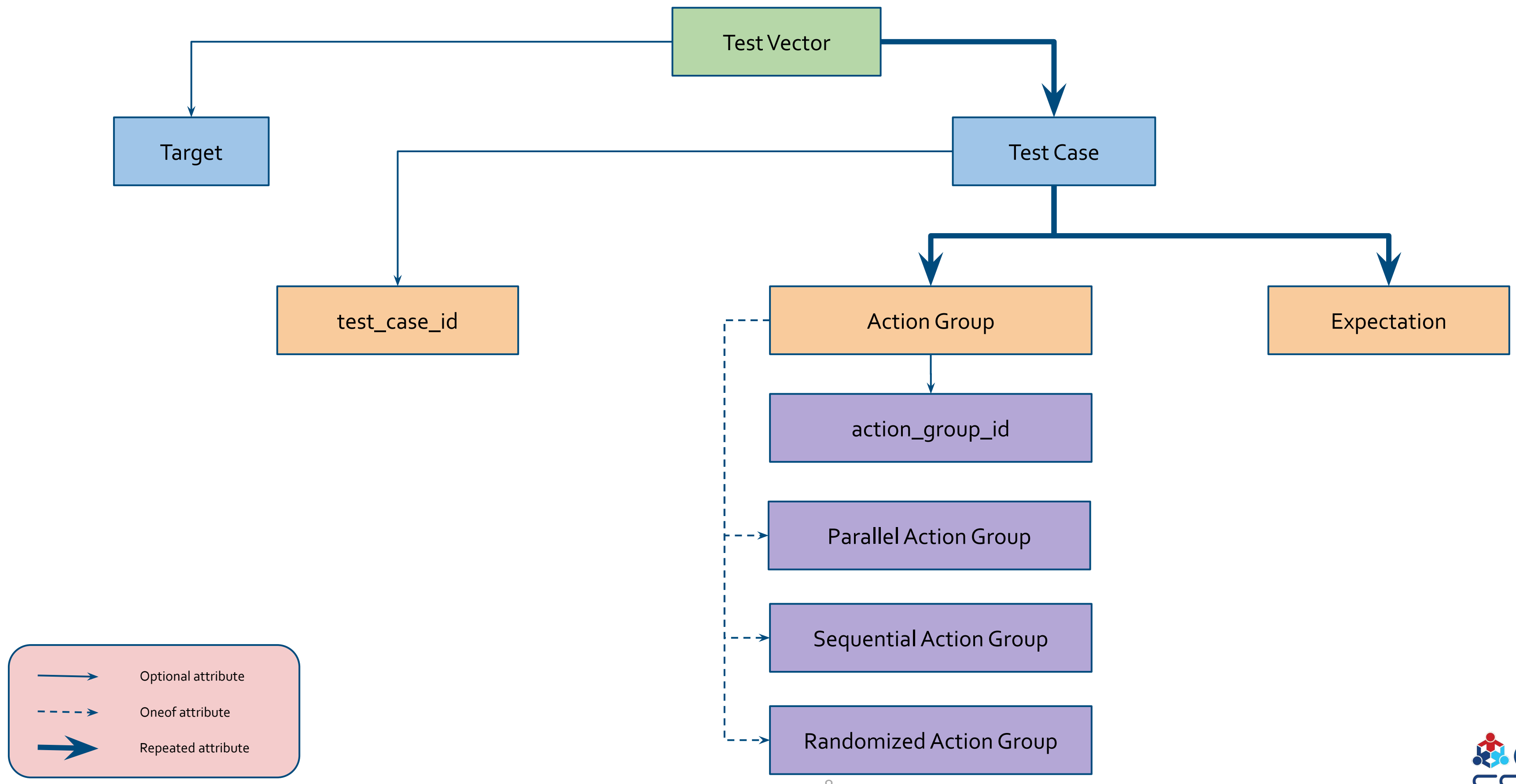
Outline

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

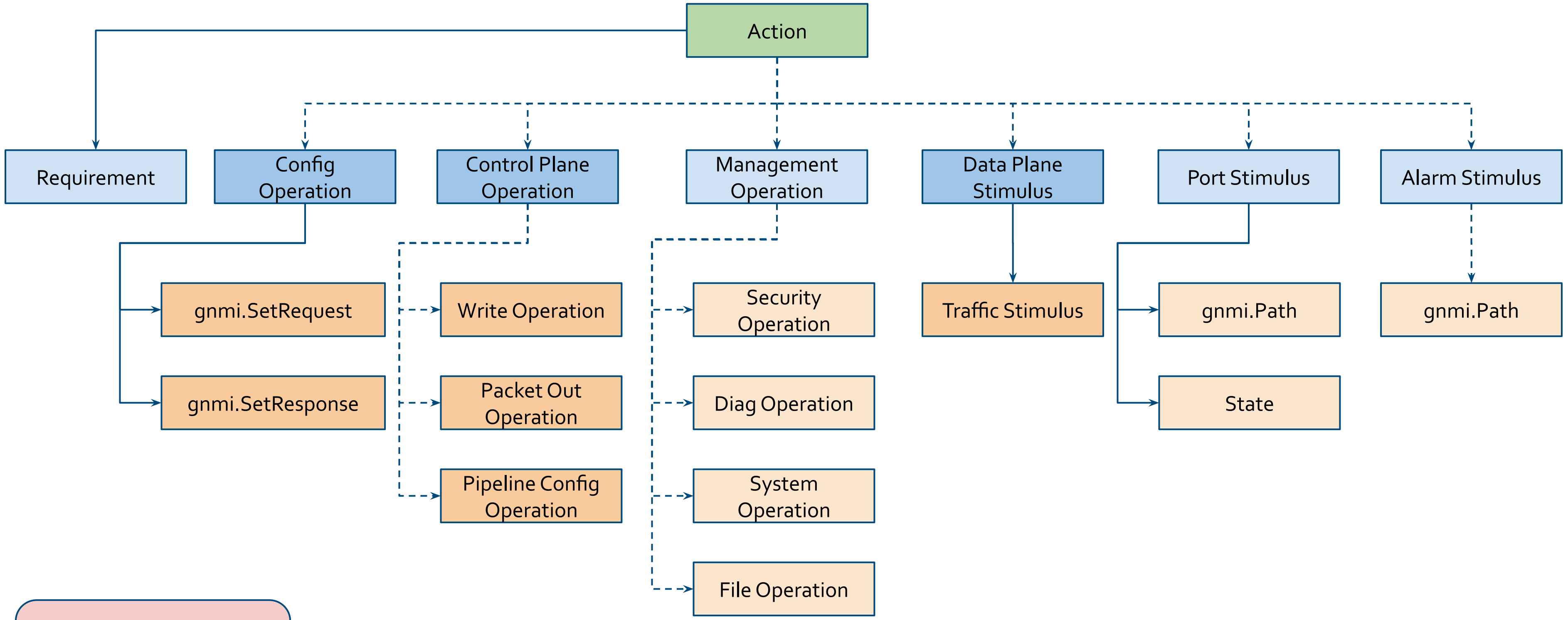
Test Vector

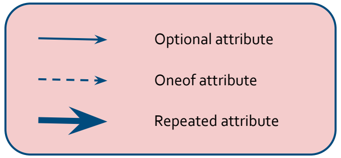
- Coded using protobufs
- TV protobuf definition is open sourced with Stratum
- gNMI, gNOI and P4Runtime also use protobufs
- Language specific source code can be generated for classes using protoc compiler

Test Vector Definition

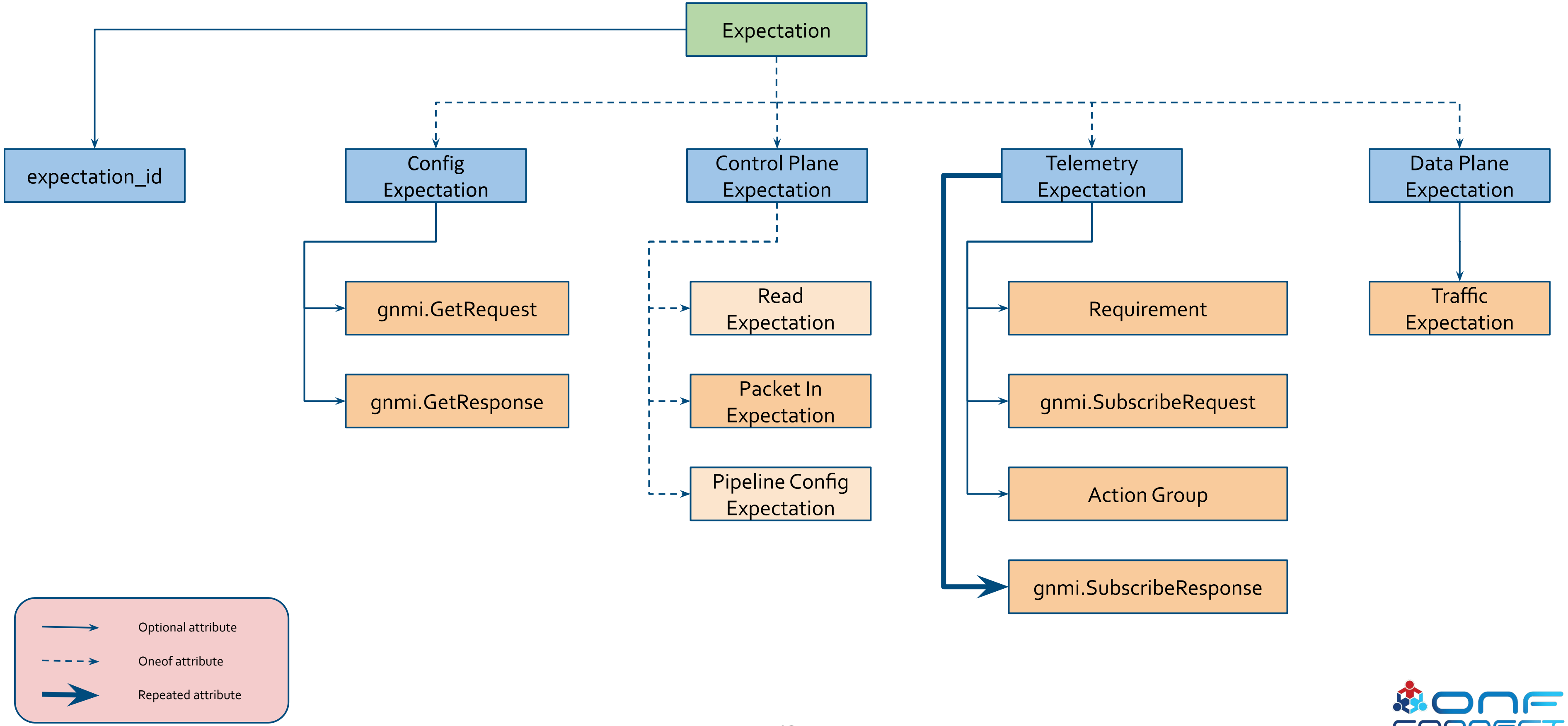


Action Definition

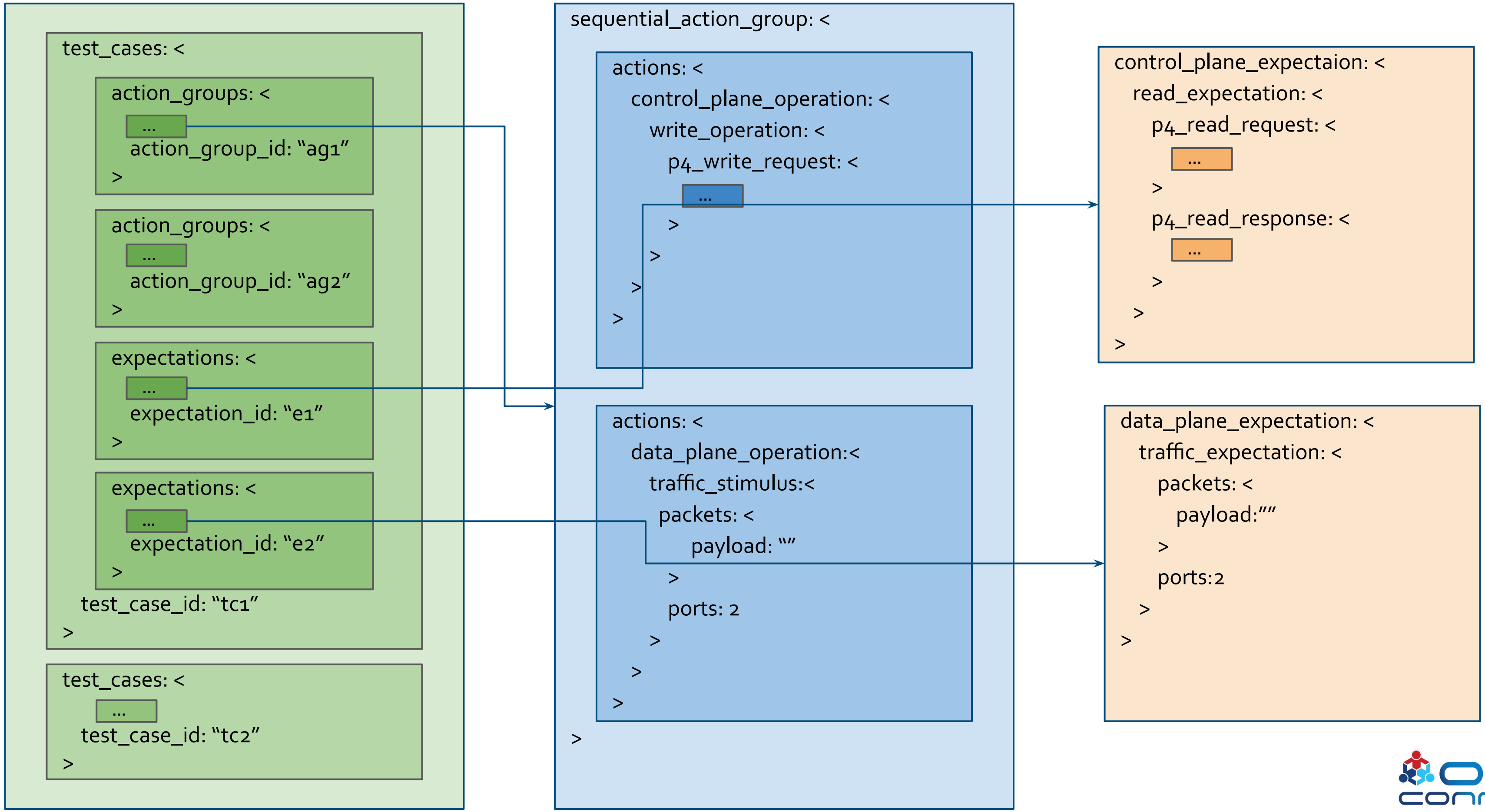




Expectation Definition



Test Vector Example



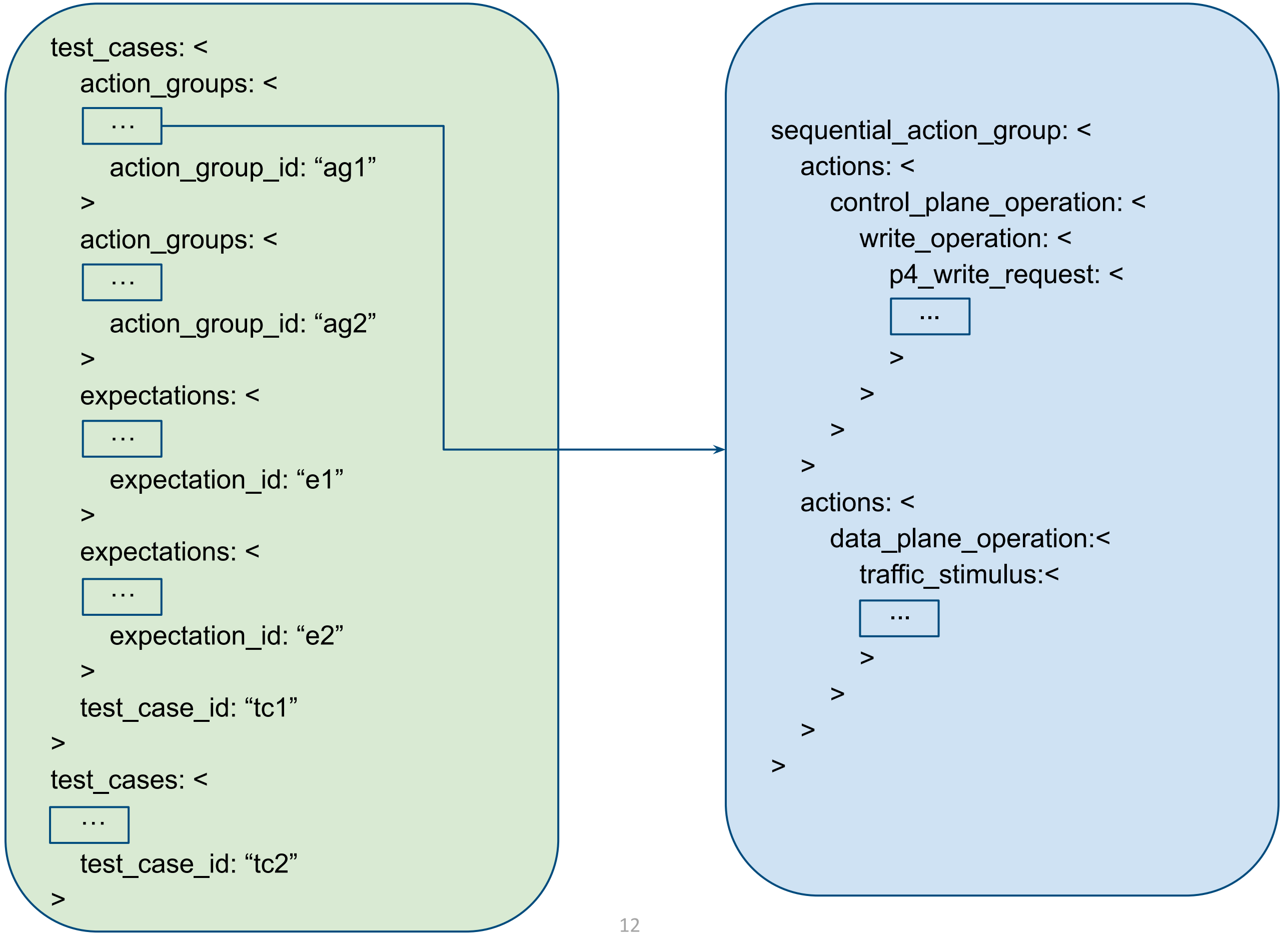
```
test_cases: <
    action_groups: <
        ...
        action_group_id: "ag1"
    >
    action_groups: <
        ...
        action_group_id: "ag2"
    >
    expectations: <
        ...
        expectation_id: "e1"
    >
    expectations: <
        ...
        expectation_id: "e2"
    >
  test_case_id: "tc1"
>
test_cases: <
    ...
  test_case_id: "tc2"
>
```

```
sequential_action_group: <
  actions: <
    control_plane_operation: <
      write_operation: <
        p4_write_request: <
          ...
        >
      >
    >
  >
  actions: <
    data_plane_operation:<
      traffic_stimulus:<
       packets: <
          payload: ""
        >
        ports: 2
      >
    >
  >
>
```

```
control_plane_expectaion: <
  read_expectation: <
    p4_read_request: <
      ...
    >
    p4_read_response: <
      ...
    >
  >
>
```

```
data_plane_expectation: <
  traffic_expectation: <
    packets: <
      payload:""
    >
    ports:2
  >
>
```

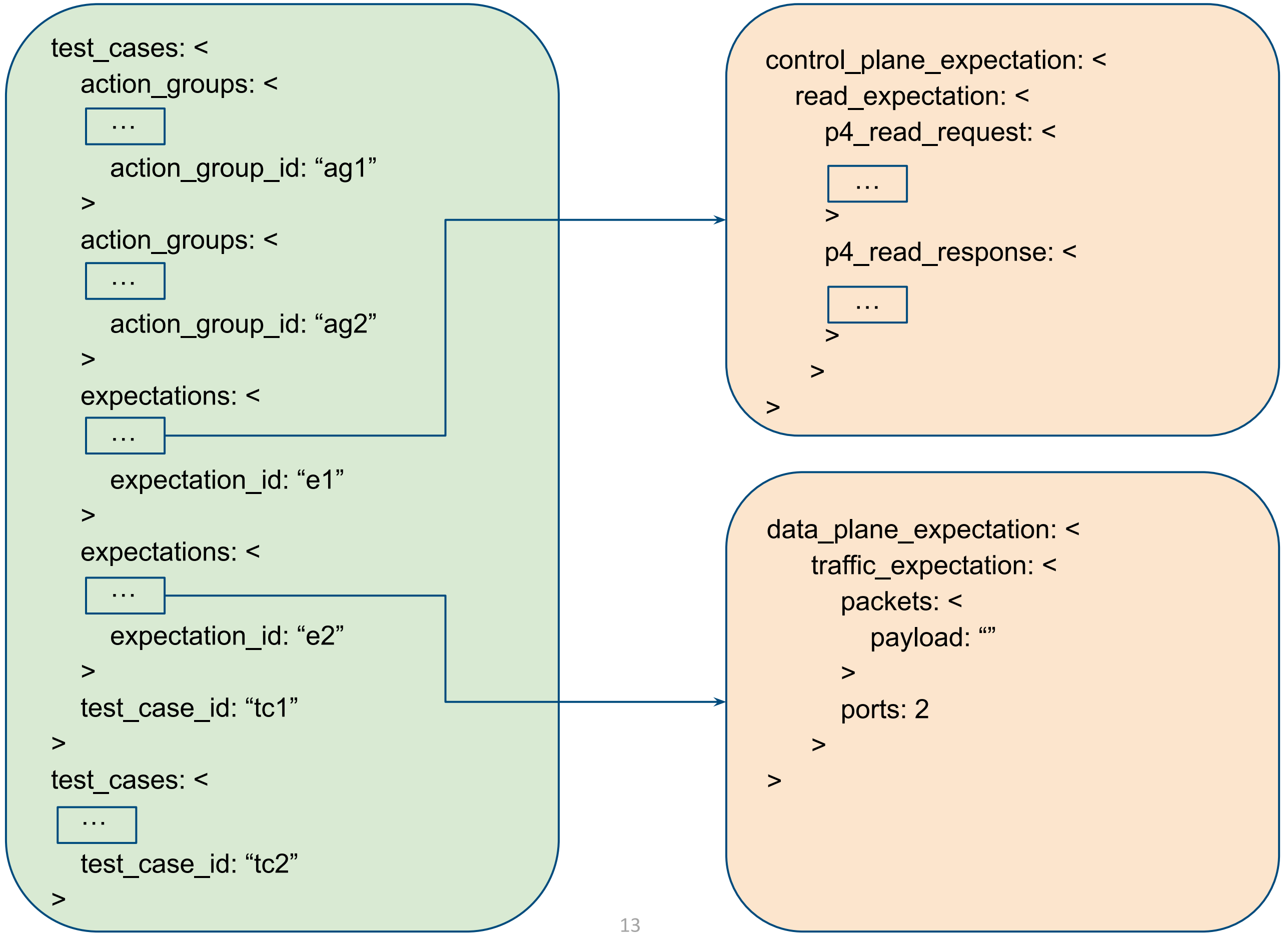
Test Vector Example



```
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: <
            ...
            >
         >
      >
   >
   actions: <
      data_plane_operation:<
         traffic_stimulus:<
         ...
         >
      >
   >
>
```

Test Vector Example

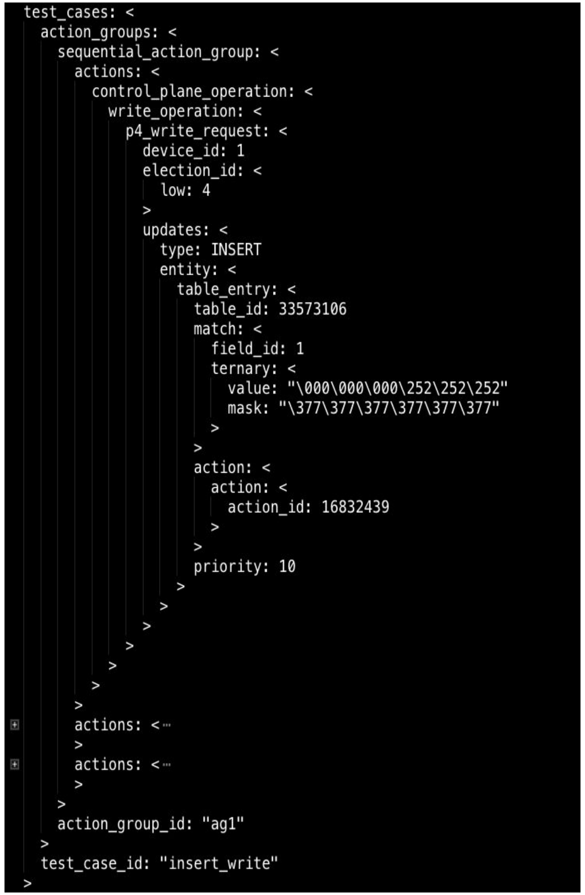


```
test_cases: <
   action_groups: <
   …
      action_group_id: "ag1"
   >
   action_groups: <
   …
      action_group_id: "ag2"
   >
   expectations: <
   …
      expectation_id: "e1"
   >
   expectations: <
   …
      expectation_id: "e2"
   >
   test_case_id: "tc1"
>
test_cases: <
…
   test_case_id: "tc2"
>
```

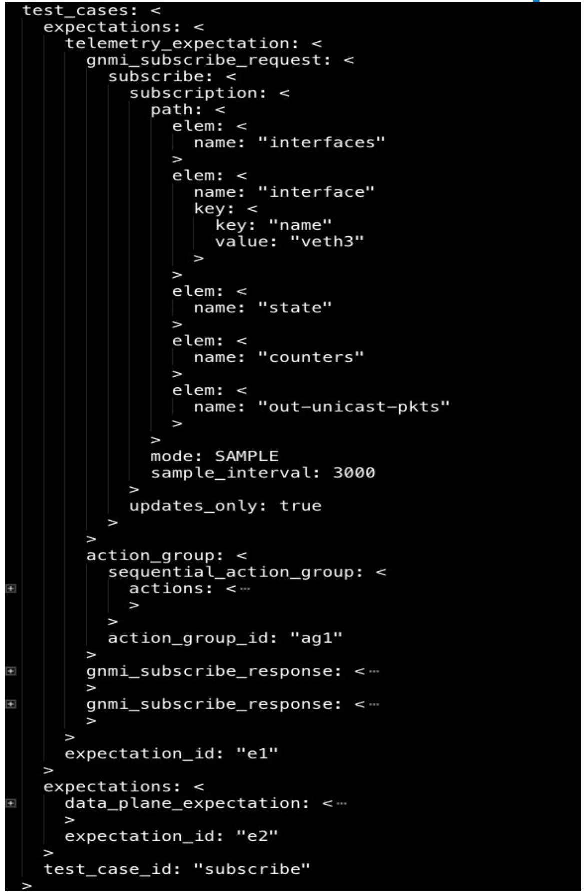
```
control_plane_expectation: <
   read_expectation: <
      p4_read_request: <
      …
      >
      p4_read_response: <
      …
      >
   >
>
```

```
data_plane_expectation: <
   traffic_expectation: <
      packets: <
         payload: ""
      >
      ports: 2
   >
>
```

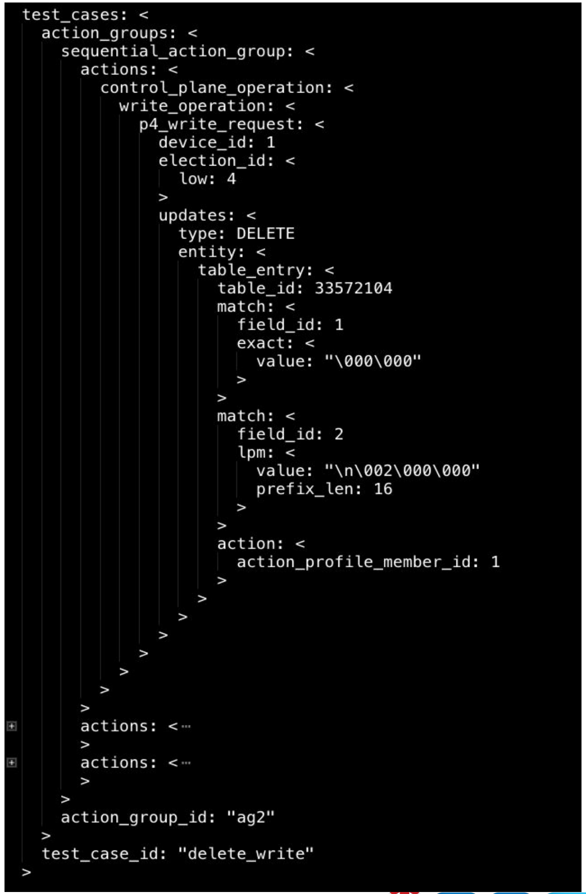
Test Vector Example



```
test_cases: <
  action_groups: <
    sequential_action_group: <
      actions: <
        control_plane_operation: <
          write_operation: <
            p4_write_request: <
              device_id: 1
              election_id: <
                low: 4
              >
              updates: <
                type: INSERT
                entity: <
                  table_entry: <
                    table_id: 33573106
                    match: <
                      field_id: 1
                      ternary: <
                        value: "\000\000\000\252\252\252"
                        mask: "\377\377\377\377\377\377"
                      >
                    >
                    action: <
                      action: <
                        action_id: 16832439
                      >
                    >
                    priority: 10
                  >
                >
              >
            >
          >
        >
      >
      actions: <…
      >
      actions: <…
      >
    >
    action_group_id: "ag1"
  >
  test_case_id: "insert_write"
>
```



```
test_cases: <
  expectations: <
    telemetry_expectation: <
      gnmi_subscribe_request: <
        subscribe: <
          subscription: <
            path: <
              elem: <
                name: "interfaces"
              >
              elem: <
                name: "interface"
                key: <
                  key: "name"
                  value: "veth3"
                >
              >
              elem: <
                name: "state"
              >
              elem: <
                name: "counters"
              >
              elem: <
                name: "out-unicast-pkts"
              >
            >
            mode: SAMPLE
            sample_interval: 3000
          >
          updates_only: true
        >
      >
      action_group: <
        sequential_action_group: <
          actions: <…
          >
        >
        action_group_id: "ag1"
      >
      gnmi_subscribe_response: <…
      >
      gnmi_subscribe_response: <…
      >
    >
    expectation_id: "e1"
  >
  expectations: <
    data_plane_expectation: <…
    >
    expectation_id: "e2"
  >
  test_case_id: "subscribe"
>
```



```
test_cases: <
  action_groups: <
    sequential_action_group: <
      actions: <
        control_plane_operation: <
          write_operation: <
            p4_write_request: <
              device_id: 1
              election_id: <
                low: 4
              >
              updates: <
                type: DELETE
                entity: <
                  table_entry: <
                    table_id: 33572104
                    match: <
                      field_id: 1
                      exact: <
                        value: "\000\000"
                      >
                    >
                    match: <
                      field_id: 2
                      lpm: <
                        value: "\n\002\000\000"
                        prefix_len: 16
                      >
                    >
                    action: <
                      action_profile_member_id: 1
                    >
                  >
                >
              >
            >
          >
        >
      >
      actions: <…
      >
      actions: <…
      >
    >
    action_group_id: "ag2"
  >
  test_case_id: "delete_write"
>
```

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

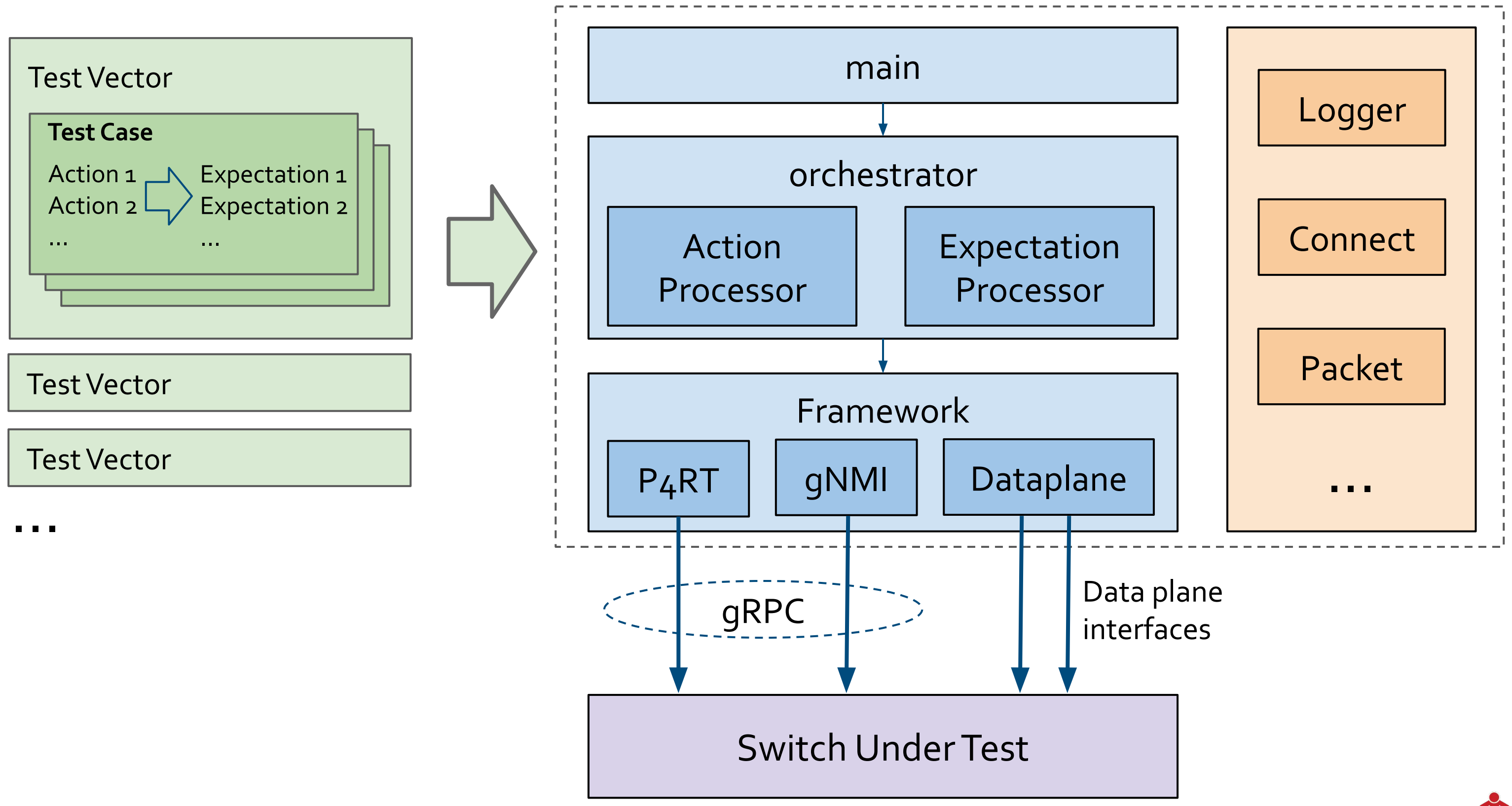
Outline

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

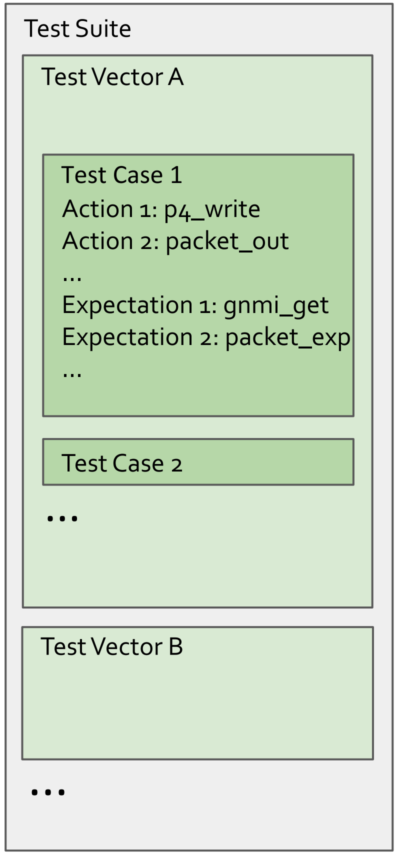
Test Vector Runner

- 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 Vector Runner Architecture

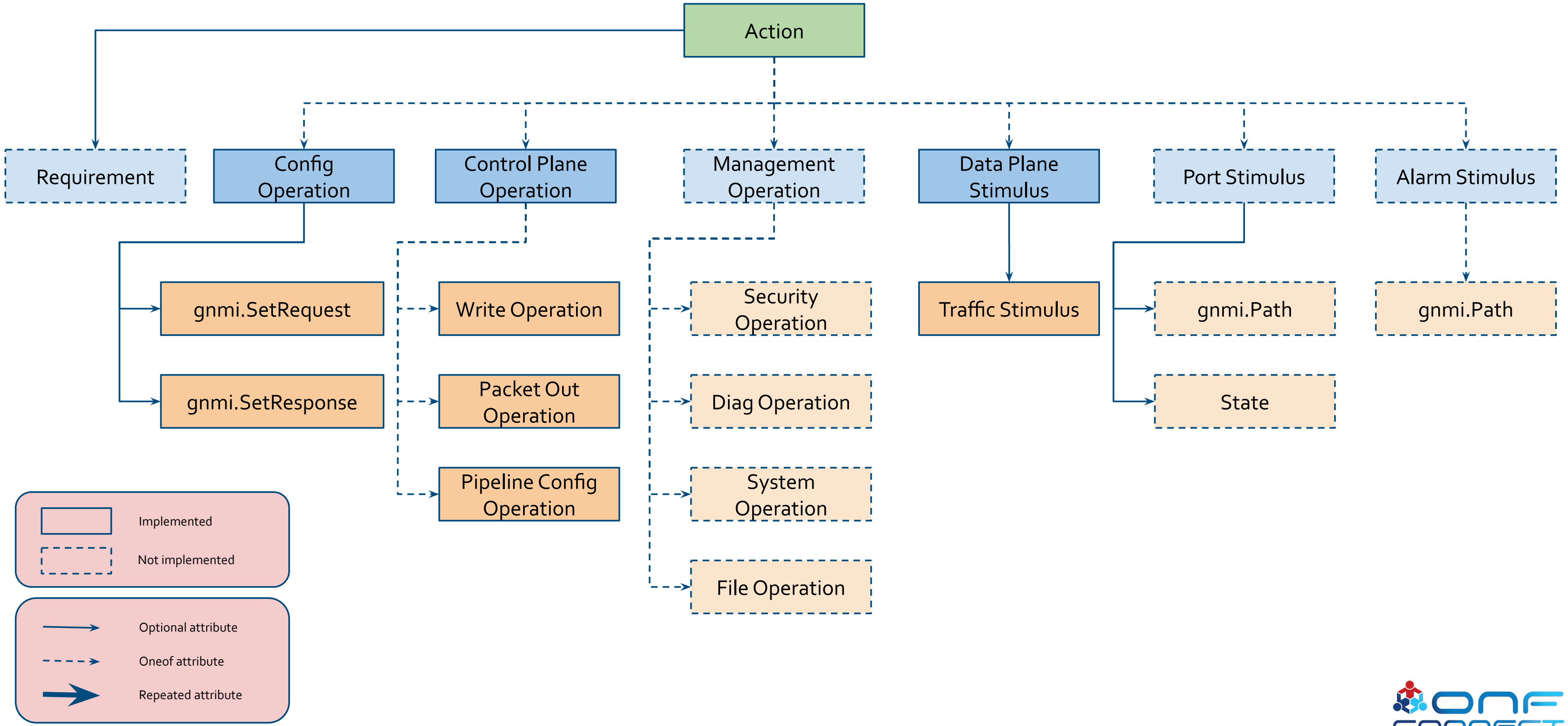


Test Vectors and Go Testing

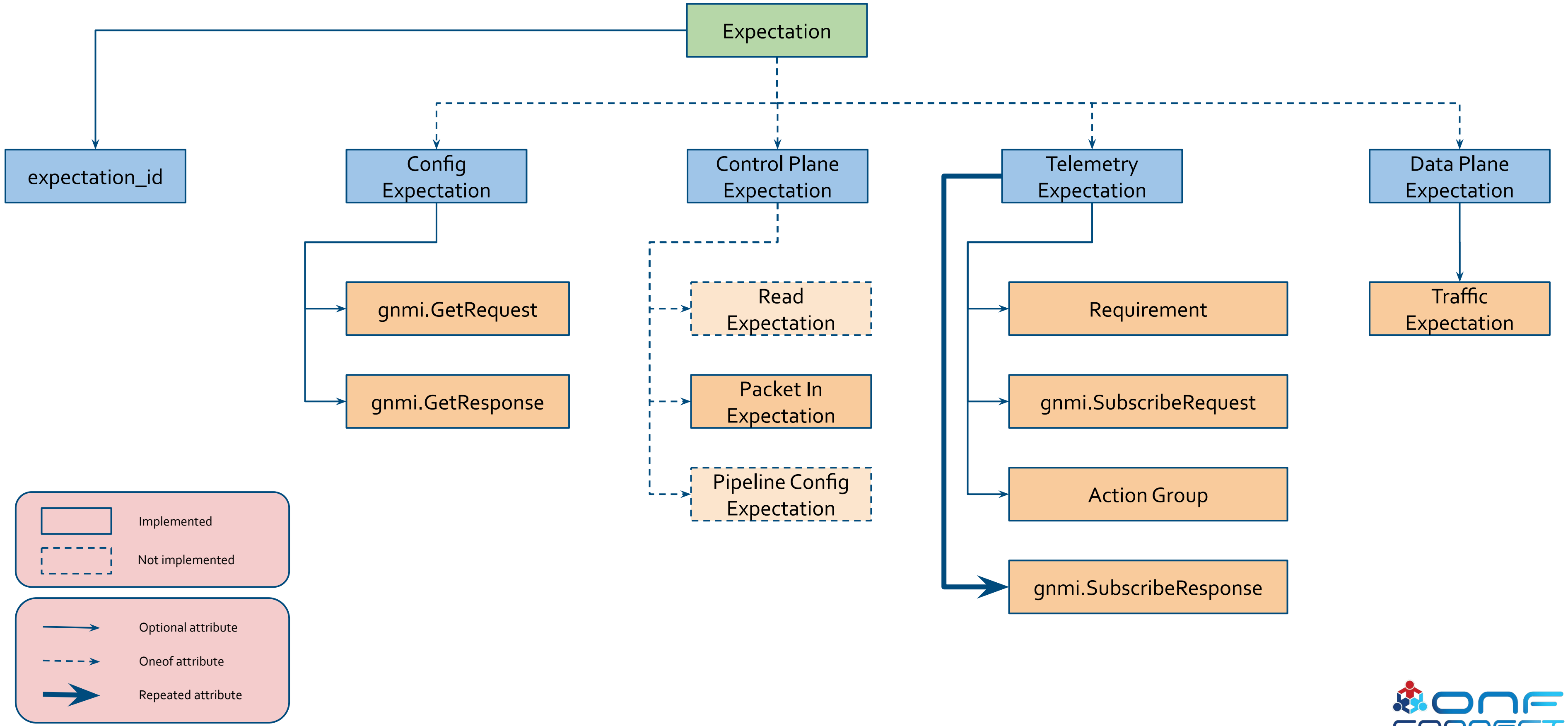


```
testing.main( []testing.InternalTest{
  testing.InternalTest{
    Name: "Test_Vector_A"
    F: func(t *testing.T) {
      t.Run("Test_Case_1", func(t *testing.T) {
        ProcessP4WriteRequest(request)
        ProcessPacketOutOperation(request)
        ...
        ProcessGnmiGetRequest(request)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

      t.Run("Test_Case_2", func(t *testing.T) {}

      ...
    }
  }
  testing.InternalTest{
    Name: "Test_Vector_B"
    F: func(t *testing.T) {}
  }
  ...
} )
```

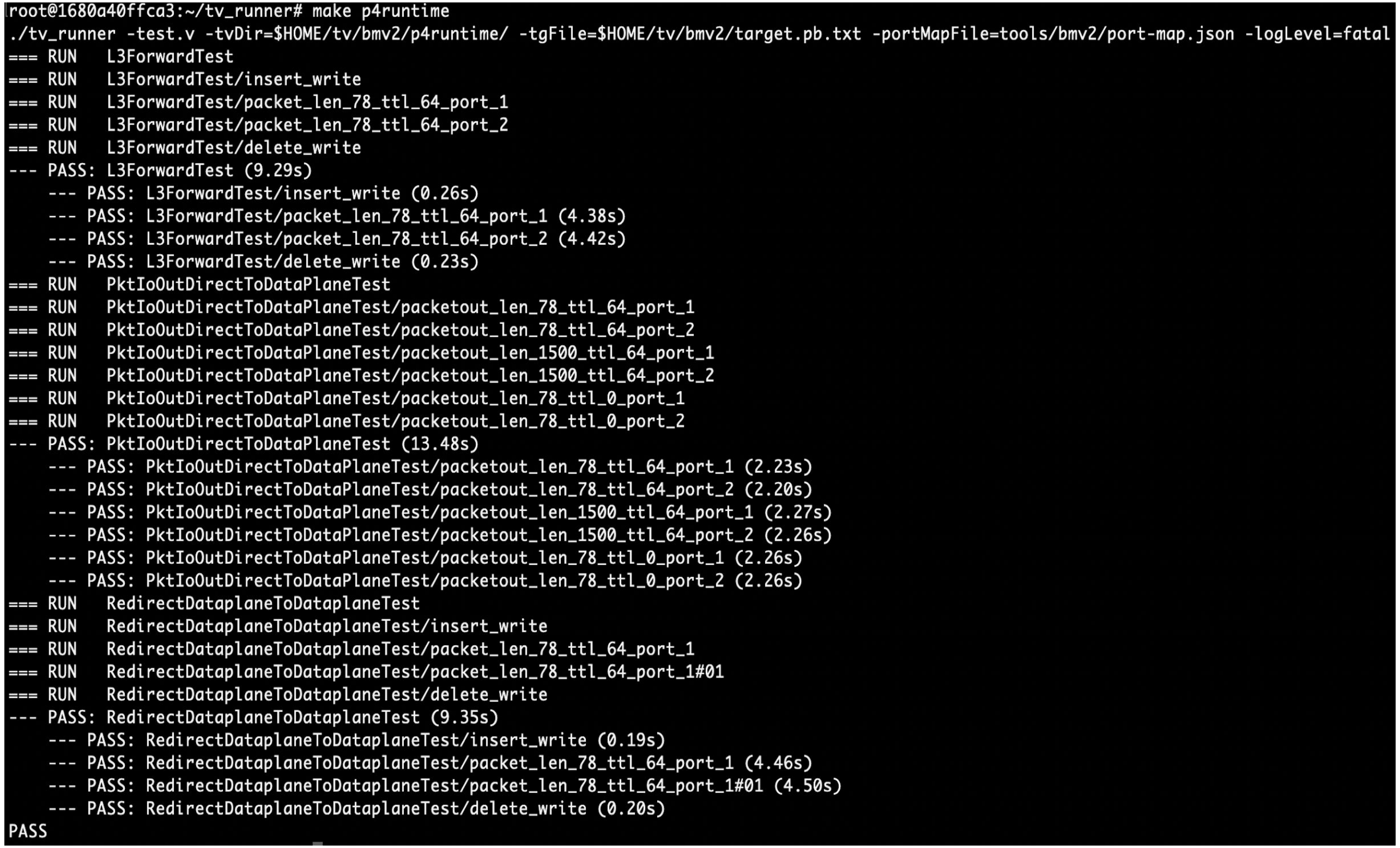
TV Runner - Actions



TV Runner - Expectations

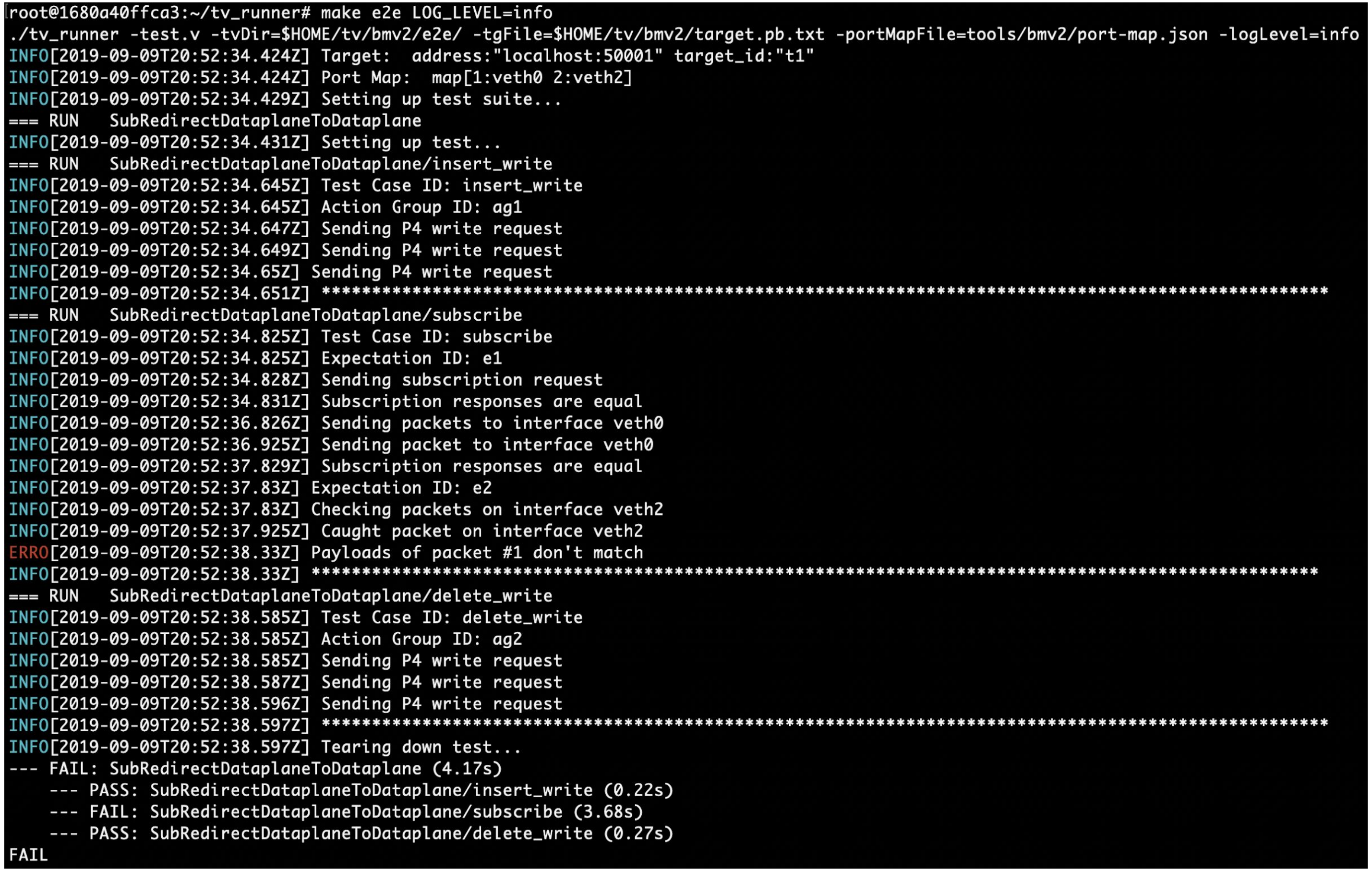


Test Execution



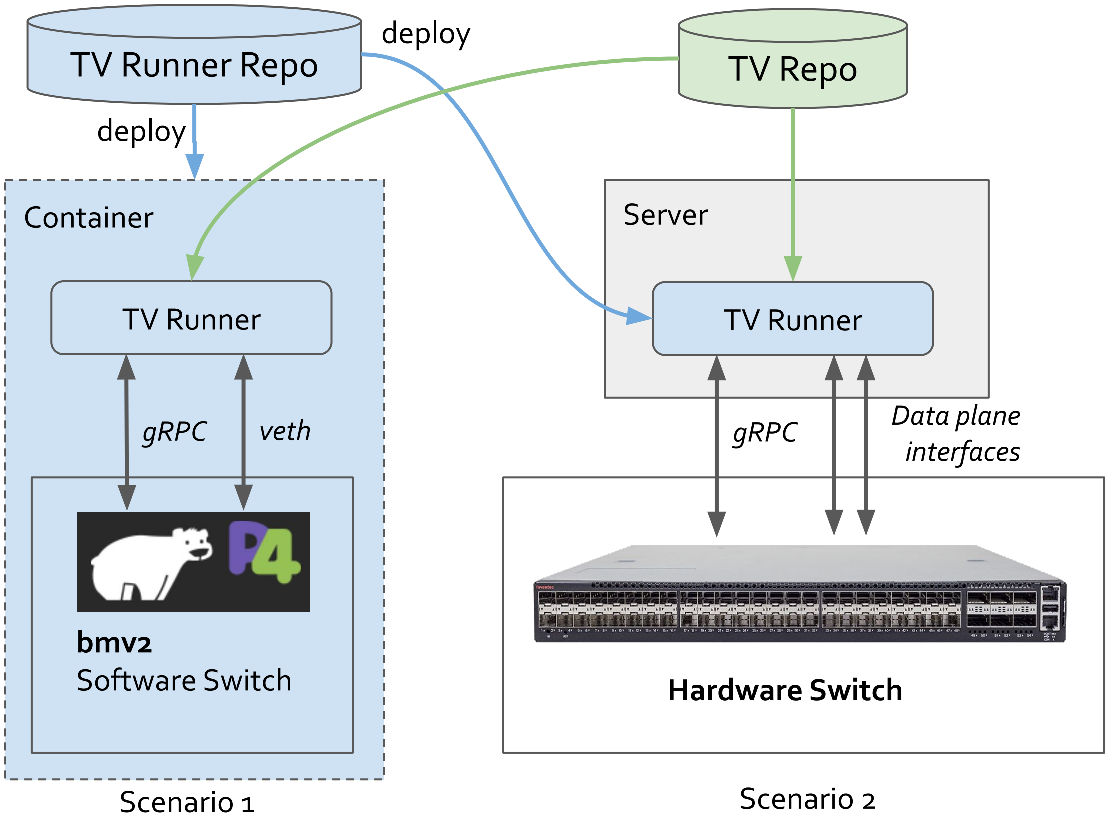
```
root@1680a40ffca3:~/tv_runner# make p4runtime
./tv_runner -test.v -tvDir=$HOME/tv/bmv2/p4runtime/ -tgFile=$HOME/tv/bmv2/target.pb.txt -portMapFile=tools/bmv2/port-map.json -logLevel=fatal
=== RUN   L3ForwardTest
=== RUN   L3ForwardTest/insert_write
=== RUN   L3ForwardTest/packet_len_78_ttl_64_port_1
=== 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 (2.20s)
    --- 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)
PASS
```

Test Execution



```
root@1680a40ffca3:~/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[2019-09-09T20:52:34.424Z] Port Map:  map[1:veth0 2:veth2]
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/insert_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[2019-09-09T20:52:34.65Z] 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[2019-09-09T20:52:37.83Z] Expectation ID: e2
INFO[2019-09-09T20:52:37.83Z] Checking packets on interface veth2
INFO[2019-09-09T20:52:37.925Z] Caught packet on interface veth2
ERRO[2019-09-09T20:52:38.33Z] Payloads of packet #1 don't match
INFO[2019-09-09T20:52:38.33Z] ****************************************************************************************************
=== RUN   SubRedirectDataplaneToDataplane/delete_write
INFO[2019-09-09T20:52:38.585Z] Test Case ID: delete_write
INFO[2019-09-09T20:52:38.585Z] Action Group ID: ag2
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



Outline

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

Test Vector Generation - Current Approach

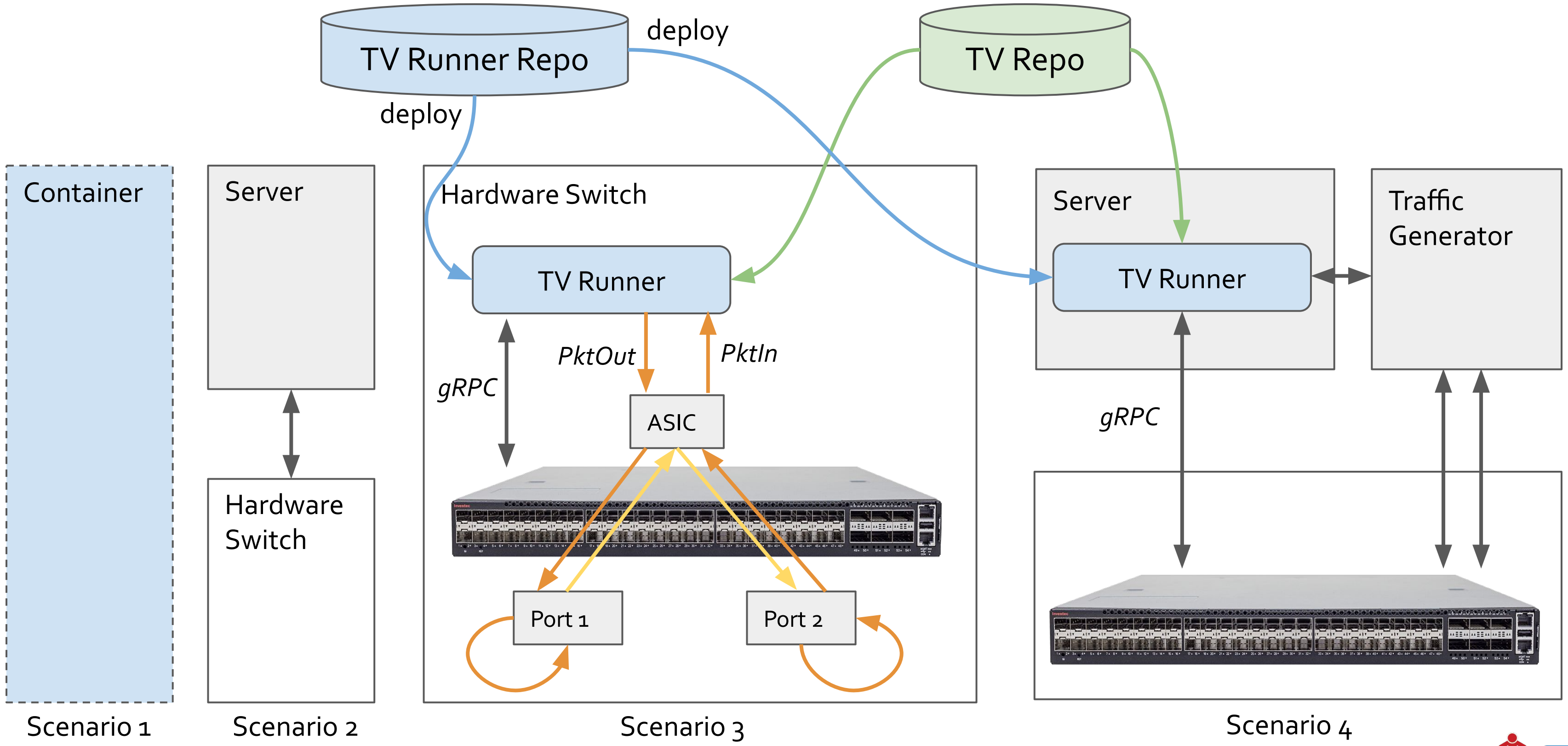
- Hand written test vectors
  - Tedious
  - Time consuming
  - Error prone
  - Hard to debug
- Semi automatically generated
  - P4RT write requests from stratum log
  - Pipeline config from P4RT generated binaries and json files
  - gNMI get operations using list of paths

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



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