SAI Test Cases

**Revision History**

| **Revision** | **Author** | **Date** | **Description** |
| --- | --- | --- | --- |
| 1.0 | Microsoft | 6/1/2018 | Initial Version |

**Overview**

This document captures an extended list of test-cases that is currently not covered in the SAI1.0 PTF test frame work. This list is not exhaustive, and more test cases would be added in future to validate the scenarios or use-cases that are not covered currently.

Ref: <https://github.com/opencomputeproject/SAI/tree/master/test/saithrift/tests>

| **ID** | **Test Category** | **Number of Test cases** | **Owned by** | **Comments** |
| --- | --- | --- | --- | --- |
| 1 | L3 Directed Broadcast | 2 |  |  |
| 2 | L3 Loopback subnet | 1 |  |  |
| 3 | MTU | 2 |  |  |
| 4 | IPv4 /32 route and same Neighbor IP | 3 |  |  |
| 5 | Dual/Triple tagging | 1 |  |  |
| 6 | MAC move | 3 |  |  |
| 7 | Neighbor MAC change and age-out | 4 |  |  |
| 8 | Traffic forwarding and FDB learning on Bridgeport | 3 |  |  |
| 9 | LAG/ECMP Hash seed | 2 |  |  |
| 10 | IPv4/v6 ECMP Group member | 5 |  |  |
| 11 | ACL Table/Group and different bindings | 6 |  |  |

**Testcase List**

The following are the list of test cases with brief description and an expected output which must conform to the behavior specified by the SAI pipeline model. The format is same as the existing SAI test list document ([link](https://github.com/opencomputeproject/SAI/tree/master/test/saithrift/tests)). Some of the test case may have few additional scenarios following the previous test-case and are captured as (I, II, III) etc.

1. L3DirectedBroadcast - I

Send IPv4 packet to a broadcast IP. The packet must have unicast destination MAC (Gateway MAC) and a broadcast destination IP (like 192.168.0.255). Verify that the packet is sent out on all member ports of VLAN if the outgoing interface is a VLAN RIF.

1. L3DirectedBroadcast – II (VLAN checks)

After executing “L3DirectedBroadcast – I”, add and remove ports to the VLAN. When a port is added, the L3 broadcast traffic must also be forwarded on the newly added member port and similar test to be executed by removing existing member port from VLAN.

1. L3LpbkSubnetTest

Send traffic on a layer 3 router interface destined to an IPv4 address within the same subnet. Have the nhop entry learnt on the same interface as incoming interface. Verify that packet is send out on the same outgoing interface as the incoming interface. Set the SAI\_ROUTER\_INTERFACE\_ATTR\_LOOPBACK\_ENABLE attribute to \_*false*\_ and verify that packet is \_*not*\_ send out on the same outgoing RIF as incoming RIF.

1. L3MtuTest

Send L3 destined packet with max interface MTU and verify forwarding. With L3 destined packet, verifying the forwarding behavior also ensures IP MTU value configured for the router interface. Increase the packet size beyond the configured MTU value and verify the behavior. Default action is to drop packets exceeding configured MTU

1. L2MtuTest

Send untagged packet with max MTU size and verify forwarding. Add VLAN tag so that the total packet size exceeds the MTU value and verify behavior. Default action is to drop packets exceeding configured MTU value.

1. L3IPv4/32Test - I

Add a route entry with /32 prefix (e.g. 10.1.1.1/32 via NH1) and a neighbor entry for the same IP address (e.g. 10.1.1.1 MAC1 Port1). Send traffic and verify that the packet takes the path specified by the /32 route entry and \_*not*\_ the neighbor path.

1. L3IPv4/32Test - II (Route delete)

After executing “L3IPv4/32Test – I”, delete the route entry with /32. Ensure the packet must be now be forwarded using the neighbor/host entry.

1. L3IPv4/32Test – III (Route re-add)

After executing “L3IPv4/32Test – II”, re-add the same route entry with /32. Ensure that the packet now take the path as specified by the route and \_*not\_* by the previously programmed neighbor entry path.

1. L2TripleTaggedTest

Create VLAN, e.g. 1000 with two-member ports. Configure port 1 as tagged member and port 2 as untagged member.

Send triple tagged packet on the tagged member port 1, with outer\_tag as VLAN - 1000, Ethertype - 0x8100 and second inner\_tag as VLAN – X, Ethertype – 0x88a8 and third inner\_tag as VLAN – Y, Ethertype – 0x8100. Verify that the packet when forwarded on the untagged port 2, it strips off the outer\_tag with VLAN – 1000 and has only two tags with Ethertypes intact.

Send dual tagged packet on the untagged member port 2, with outer\_tag as VLAN – X, Ethertype – 0x88a8 and inner\_tag as VLAN – Y, Ethertype – 0x8100. Verify that the packet when forwarded on the tagged port 1, it adds an outer\_tag with VLAN – 1000, Ethertype – 0x8100 and send as triple tagged with inner Etherypes intact.

1. L2MacMoveTest – I (Station Movement)

Create VLAN, e.g. 100 and add member ports 1, 2 and 3. Send packet on port 1 with src\_mac=MAC1 and dst\_mac= MAC2. Verify MAC1 is learnt on port 1 and packet is flooded to other member ports (2 and 3 in this example).

Send packet on port 2 with src\_mac=MAC2 and dst\_mac= MAC1. Verify MAC2 is learnt on port 2. After learning, verify that packet from port 1 is only forwarded to port 2 and not to port 3.

Repeat the test by sending same packet (src\_mac=MAC2 and dst\_mac= MAC1), on port 3. Verify that station-movement occurred and MAC2 is learnt on port 3. Packet from port 1 destined to MAC2 must be forwarded to port 3 and not to port 2 after the MAC-movement.

1. L2MacMoveTest – II (With IPv4 Neighbor entry)

Create a neighbor entry with MAC1. Let MAC1 be learnt on port 1 and observe traffic forwarding. Execute “L2MacMoveTest” and let MAC1 be learnt on 2nd port. Verify L3 forwarding and ensure that traffic is now forwarded to the newly learnt port.

1. L2MacMoveTest – II (With IPv6 Neighbor entry)

Same as "L2MacMoveTest – II” but for IPv6 neighbor.

1. L3IPv4NeighborMacTest

Create two VLAN router interfaces in the same VRF. Create nhop, route and neighbor entry for destination. Send packet on one router interface and verify packet received on the other router interface. Simulate a MAC address change for the neighbor entry and verify traffic. The packet must now be forwarded with the new MAC address as destination MAC.

1. L3IPv6NeighborMacTest

Same as “L3IPv4NeighborMacTest” but for IPv6 neighbor

1. L3IPv4NeighborFdbAgeoutTest

Create two VLAN router interfaces in same VRF. Create nhop and route for destination. Learn ARP for the nhop and check FDB table for the MAC learnt from the ARP packet in that VLAN. Simulate FDB age-out by clearing the FDB entry or wait for age-out time. After the FDB entry is cleared, the layer 3 traffic via this nhop must take the action as specified by SAI switch attribute SAI\_SWITCH\_ATTR\_FDB\_UNICAST\_MISS\_PACKET\_ACTION.

1. L3IPv6NeighborFdbAgeoutTest

Same as “L3IPv4NeighborFdbAgeoutTest” for IPv6 neighbor by clearing/deleting the learnt FDB entry for the IPv6 neighbor

1. BridgePortTest - I

Create a bridge-port and verify traffic forwarding. Disable bridge-port admin state and verify packets are dropped.

1. BridgePortTest – II

After “BridgePortTest – I”, flush the FDB entries learnt on the bridge-port. Verify that all the entries are cleared, and new MAC learning is not happening.

1. BridgePortTest – III

After “BridgePortTest – II”, enable bridge-port. Verify that traffic forwarding is restored, and MAC learning is happening.

1. LAGHashseedTest

Create a LAG group with 4 ports 1 through 4. Setup static FDB entries for the LAG and send packet to this destination MAC address. Send 100 packets with varying 5-tuple and check order/sequence of the distribution of packets received on ports 1 through 4. Change the LAG Hash seed value to 10 and compare the order/sequence of the distribution of packets received for the same set of 100 packets on ports 1 through 4. Verify that it is different after changing the hash seed.

1. L3IPv4EcmpHashseedTest

Create a VRF with IPv4 and IPv6 enabled. Create 4 router interfaces in the same VRF. Create a route (/24 mask) with nhop and neighbor entry in three router interfaces. Send 100 streams with varying 5-tuple combinations to the destination IP on one port and verify distribution on the three router interfaces for which the nhops are present. Change the ECMP hash seed value to 10 and verify distribution. [Same as “LAGHashseedTest”]

1. L3IPv4EcmpGroupMemberTest – I

Create a VRF with IPv4 and IPv6 enabled. Create three router interfaces in the same VRF. Create a route (/24 mask) with nhop and neighbor entry in two router interfaces. Send ~10,000 streams with varying 5-tuple combinations to the destination IP on one port and verify distribution on the two router interfaces for which the nhops are present. Create another router interface and add nhop and neighbor entry over this interface to the route configured above. NH group object shall be updated and verify traffic distribution to the newly added nhop.

1. L3IPv4EcmpGroupMemberTest – II

After executing “L3IPv4EcmpGroupMemberTest – I”, remove the nhop entry on router interface 1 from the route while traffic flow is still active. Verify the traffic is not forwarded to removed nhop and distribution happens on the remaining set of nhops.

1. L3IPv4EcmpGroupMemberTest – III (Zero member)

Same as “L3IPv4EcmpGroupMemberTest” but the ECMP group object to which the route is pointing has *zero* members. Verify that traffic is dropped

1. L3IPv6EcmpGroupMemberTest – I

Same as “L3IPv4EcmpGroupMemberTest – I” for IPv6 destination

1. L3IPv6EcmpGroupMemberTest – II

Same as “L3IPv4EcmpGroupMemberTest – II” for IPv6 destination

1. L3AclTableTest – I (LAG Binding)

Create two router interfaces with one over a LAG and another VLAN. Create a nhop, route and neighbor entry for destination and verify traffic forwarding. Create ACL table with stage Ingress, destination IPv4 and bind point type LAG. Create an ACL entry for the destination IP with action \_*drop*\_ and verify traffic.

1. L3AclTableTest – II (VLAN Binding)

Same as “L3AclTableTest – I” for bind point type VLAN. Create ACL entry for the traffic over VLAN in this case.

1. L3AclTableTest – III

After “L3AclTableTest – II”, bind the same ACL table to a port. Test is to verify dynamically changing the bind point type of an ACL table from VLAN to a PORT. Verify that the bind point can be changed, and the ACL action is only applied on the associated port but not the VLAN.

Remove all the table bindings and verify that traffic is now forwarded for the LAG, VLAN and PORT

1. L3AclTableGroupTest – I (LAG Binding)

Create route objects as in “L3AclTableTest – I”. Create two ACL tables with stage ingress, and one with destination IPv4 and other with source IPv4. Create ACL entry in both the tables with source IP address action drop and destination IP address action drop. Associate the two tables to an ACL group with type “*parallel*” and bind it to the LAG. Send traffic matching the destination IP and source IP and verify output

1. L3AclTableGroupTest – II (VLAN Binding)

Same as “L3AclTableGroupTest – I” for bind point type VLAN.

1. L3AclTableGroupTest – III

Remove the group binding and verify that traffic is now forwarded for both the LAG and VLAN.