ECMP Convergence Acceleration Test Cases

Table of Contents

[1 Document History](#_Toc508231556)

[2 Abbreviations](#_Toc508231557)

[3 References](#_Toc508231558)

[4 Problem Overview](#_Toc508231559)

[5 Proposed Solutions](#_Toc508231560)

[5.1 BGP next hop tracking](#_Toc508231561)

[5.2 Removing the ECMP path](#_Toc508231562)

[6 Scale / Performance](#_Toc508231563)

[7 CLI](#_Toc508231564)

[8 Related SAI APIs](#_Toc508231565)

[9 Test Setup](#_Toc508231566)

[10 Test Cases](#_Toc508231567)

[11 Test Case Execution](#_Toc508231568)

[12 Test Report](#_Toc508231569)

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 0.1 | 05/03/2018 | Harsha | Initial version |
| 1 |  |  |  |

# Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| BGP | Border Gateway Protocol |
| ECMP | Equal Cost Multi Path |
| IGP | Interior Gateway Protocol |
| RIB | Routing Information Base |
| ASW | Access Switch |
| CSW | Core Switch |
| FAB | Fabric Switch |

# References

|  |  |
| --- | --- |
| **Type** | **Link** |
| Design document | <https://github.com/Azure/SONiC/blob/gh-pages/doc/sonic-ecmp-acceleration.docx> |

# Problem Overview

It is standard practice in today’s networks to distribute traffic across multiple paths. This is achieved either statically or dynamically with protocols such as BGP, or IGPs. By default, BGP will choose only one path during its bestpath selection but in today’s data centers, network nodes are connected to multiple other nodes with the intend to provide increased bandwidth as well as multiple equal cost paths for redundancy purposes and load balancing. As such, BGP is configured to take advantage of the multiple available paths and selects them, if available, for a prefix during bestpath calculation for installation into the RIB and forwarding.

When one of those paths becomes unavailable due to an interface going down or administratively shut, BGP will either detect that or be notified of changes in its next hops. It will then determine which paths are affected by this change, usually via a walk of the prefix table, and for each prefix it will rerun its bestpath selection algorithm and modify the ecmp paths calculated for that prefix accordingly. It will then proceed downloading all the affected prefixes into RIB which in turn it will download into sonic.

This process is not instantaneous and until BGP converges from such an event and the hardware is updated, forwarding is still using the unavailable path which results in black holing traffic partially for any prefix that makes use of that path.

It is therefore desirable to reduce significantly the amount of time traffic is being dropped over the unavailable path and accelerate at a lower level the ecmp groups convergence by removing from forwarding any paths being affected due to the interface state change.

# Proposed Solutions

Before outlining the two proposed solutions, it is important to understand the different ways BGP may track next hop changes and how it reacts to them. This will provide more insight as to what advantages and disadvantages each solution has and decide what is best suited for the current sonic architecture and data center deployments as well as what may be desirable in the future.

## BGP next hop tracking

Different BGP implementations, may have different ways of tracking and validating BGP next hops.

Older BGP implementations are using a polling/periodic mechanism. In those cases, BGP uses a timer and upon timer expiration, the RIB is polled and next hops are validated. In between walks, next hop changes due to an interface flap that doesn’t affect a BGP session, are not detected. Similarly, between scan cycles, network failures are not detected and routing loops or traffic loss may occur.

Newer BGP implementations, are using an event driven mechanism for next hop address tracking. Usually this involves BGP registering with the RIB the prefixes it needs RIB to track. Next hop changes are rapidly reported to BGP for the registered prefixes as they are updated in the RIB. This event driven optimization improves overall BGP convergence by reducing the response time to next hop changes for routes installed in the RIB.

## Removing the ECMP path

The simplest solution to solve the problem from implementation’s perspective, is to perform the following steps **inline** in the orachagent when the i/f down state change is detected:

* Obtain the alias of the interface.
* Walk all nexthops to see which ones go over the affected interface.
* Mark any affected nexthops.
* For each affected nexthop go through the ecmp groups and see if there is an affected member.
* If there is, call the remove\_next\_hop\_group\_member SAI API to remove the path from the ecmp group in the SAI db.

The steps are similar on i/f state up:

* Obtain the alias of the interface.
* Walk all nexthops to see which ones go over the affected interface.
* Clear flag of any affected nexthops.
* For each affected nexthop go through the ecmp groups and see if there is an affected member.
* If there is, call the create\_next\_hop\_group\_member SAI API to remove the path from the ecmp group in the SAI db.

This approach is preferable than the one removing the ecmp member from the group at the orchagent level.

# Scale / Performance

No scale / performance test involved in this test plan.

# CLI

No additional CLIs have been added to support this feature.

# Related SAI APIs

N/A

# Test Setup

The following topology has been used to test and validate the new functionality:

**FAB VM0100**

**100.1.1.1**

16 ECMP paths available on DUT for 100.1.1.1

**ASW VM0130**

**ASW VM0131**

**FAB VM0114 100.1.1.1**

**FAB VM0115 100.1.1.1**

**DUT**

**FAB VM0101 100.1.1.1**

BGP

BGP

BGP

BGP

BGP

BGP

BGP

BGP

**ASW VM0116**

**ASW VM0117**

As shown in the above diagram, this test suite makes use of T1 topology that provides two sets of VMs:

* One set of 16 VMs acting as ASWs (bottom ones)
* Another set acting as FABs (top ones)

In our case, DUT is performing as a CSW, and it connects to 16 FAB VMs, and another 16 ASW VMs.

The DUT is receiving 6403 routes from each of BGP neighbour (FAB VMs), and traffic is sent from ASW VM (ASW VM0116 in our case).

# Test Cases

The following are the list of test cases with brief description and an expected output.

* **Test Case #1**: Clear the counters on the DUT. Start the traffic from one of the ASW VMs (this test case sends 250 packets per second for 2 minutes). Then shut down the interface on which traffic is flowing. Verify that the interface status is correctly shown as ‘down’ on the DUT. Also verify that this particular path has been removed from the prefix. Also verify that convergence time is 1 second or less. This test also checks that, after shutting down, relevant syslog messages are logged. **Result: PASS**
* **Test Case #2**: Perform the above test going from say, interface 4🡪3 ECMP path for the corner case. **Result: PASS**
* **Test Case #3**: Continue removing paths in the same way as in the above test cases, and verify that convergence time is 1 second or less until one path is left.  **Result: PASS**
* **Test Case #4**: Remove the remaining path. Check that SAI has removed all the ECMP groups that have been formed. **Result: PASS**
* **Test Case #5**: Bring the interfaces back up, and verify that paths are added back to the prefix. Also verify that the interface status is correctly show as ‘up’ on the DUT. There should be no traffic loss. **Result: PASS**

# Test Case Execution

A new yml file ecmp.yml has been created. This takes care of setting up VMs with required configuration to run the test cases, and also end-to-end execution of the whole test suite. No additional configuration is required.

Use “**ecmp**” as tag to run this test suite.

The command to start the test suite is:

**ansible-playbook test\_sonic\_by\_tag.yml -i inventory --limit {DUT\_NAME}, --become --tags ecmp --extra-vars “testbed\_type={TESTBED\_TYPE} vm\_hosts=[’10.0.0.200’,’10.0.0.201’,’10.0.0.202’,’10.0.0.203’] vm\_source=’10.0.0.216’”**

where

**DUT\_NAME:** Host name of the DUT under test.

**TESTBED\_TYPE:** Testbed type used for testing the feature; **‘t1’** in this case.

**vm\_hosts:** Represents the list of destination VMs to which the traffic is flowing. If we’re testing with 4 interfaces, specify 4 destination VM IPs. If we’re testing with 16 interfaces, speficy 16 VM IPs.

**vm\_source:** Represents the source VM from which traffic is sent from.

If the test case fails, the interfaces that are already brought down will be brought back up automatically. No manual cleanup is required.

# Test Report

|  |  |  |
| --- | --- | --- |
| **Test Case (Refer Section 10 for Details)** | **Test Result**  **(Image without FIB Acceleration Feature)** | **Test Result (Image with FIB Acceleration Feature)** |
| Test Case #1 | Path has been removed from the prefix.  Convergence time was between 4-6 seconds. | Path has been removed from the prefix.  Convergence time was less than 1 second.  Found “**invalidnexthopinNextHopGroup**” and “**setNextHopFlag**” in syslog. |
| Test Case #2 | Path has been removed from the prefix.  Convergence time was between 4-6 seconds. | Path has been removed from the prefix.  Convergence time was less than 1 second.  Found “**invalidnexthopinNextHopGroup**” and “**setNextHopFlag**” in syslog. |
| Test Case #3 | SAI continued to remove paths from the prefix. | SAI continued to remove paths from the prefix. |
| Test Case #4 | SAI has removed all the paths from the ECMP group. | SAI has removed all the paths from the ECMP group. |
| Test Case #5 | Paths are added back to the prefix.  No traffic loss. | Paths are added back to the prefix.  No traffic loss. |