SONiC unit test and function test enhancement

Taskin Ucpinar
EdgeCore
Edgecore SONiC Development and Testing Environment

Towards a deployment-ready SONiC
• End-to-end testing in development workflow
• Coverages and Automation
• HW platform validations
• Pre-SI RAS, Performance testing
• Sonic ecosystem integrations
Edgecore is Dedicated to Future of SONiC

AS7816-64X
AS7212-54X
AS6712-32X
AS7712-32X
AS9716-32D
AS7712-32X
AS7312-56X
AS7312-54X
AS7712-32X
AS7312-54XS
AS7716-32X
AS5712-54X
AS7312-54X
AS7712-32X
AS7312-54XS
AS7716-32X
AS7116-54X
AS7512-32X
AS7716-32XB
AS7512-32X
AS7816-64X
AS6716-32X
AS4222-28PE

TO BE ANNOUNCED DURING OCP
Future of SONiC

- SONiC Feature Set Growth
- Rapid Development Environment
- Ensure Stability/Reliability
- Testing
  - Interoperability
  - Regression
  - CI/CD
Testing - Regressions

- Getting the full picture
- Completing the coverage
- Hardening
- Community Services
Getting The Full Picture

SONiC201705
- BGP
- ECMP
- LAG
- LLDP
- QoS - ECN
- QoS - RDMA
- One Image

SONiC201709
- VLAN
- ACL Per/Den
- IPv6
- Tunnel Decap
- Mirroring
- BGP MP
## Getting The Full Picture

<table>
<thead>
<tr>
<th>Release</th>
<th>SAI Version</th>
<th>No of Features</th>
<th>No of Ansible Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONiC201705</td>
<td>0.9.4</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>SONiC201709</td>
<td>0.9.4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>SONiC201712</td>
<td>1.0</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>SONiC201803</td>
<td>1.2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>SONiC201807</td>
<td>1.3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>SONiC201811</td>
<td>1.3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>SONiC.201903</td>
<td>TBD</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>63</strong></td>
<td><strong>20</strong></td>
</tr>
</tbody>
</table>
Testing From All Angles

- Test units/components/functions/functionality
- Not only for testing, but for educational purposes
- Independent on features/platforms
- Unit Tests: Typically implemented by developers
Completing The Picture

- 100% Coverage
  - All Features with Tests
  - Coverage % Unknown
  - Stability Unknown
- Features Missing Tests
- Automation
  - Auto detect and execute
  - Regression Testing
- Hardening
  - Stability Improvements
  - Measure Coverage %
  - New tests

Edgecore Community Labs
- Multiple Community Labs Across Globe
- SONiC Devices and Packet Generators
- Available to Edgecore Partners, Customers, Community
Unit Test Software Architecture
ACL/SwSS Logical View, an Example

Use the following to replace original implement:
1. c++11 lambda
2. gmock class instance
3. member function of Fixture class of gtest
4. c function pointer

unit test of AccOrch like function test for all operations by user view under controlled environment
Thank You
Backups
for example
ACL/CLI testing

With redis

```
import config/*.py
run acl_test*.py
import sonic-pyswsssdk
redis
run
Test Result
```
How Gtest Works

Run gtest
- Download and compile gtest into static library
- Create gtest project
- Create a test case – import source code and dependent packages
- Add instrumentation code to source code to be tested
- Run the test

Run gtest for SONiC
- Test behavior before code modification
- Modify code (add code, change code such as bug fixing)
- Test behavior and compare with before
Gtest for SONiC

Our targets
- Create an unit test framework for SONiC contributors as a developing and debugging tool
- Reduce the work that contributors involve to run a test
- Easy management interface

Gtest advantages
- Run the exact same test repeatedly
- Track the context state info when hitting a bug

Constraints of Gtest on SONiC
- Use production language framework
  - Use Python/Go framework for code in languages such as Python/Go
- Typically not cover complicated operations such as send/receive packets, database operations etc.
- Also typically not cover script/shell code
Gtest levels
  Simulate referenced components or do component crossing test depending on feature/developer requirements
    Redis, SAI, socket, …

Gtest performance
  More closer to real environment, more time to run

Import source code dependencies
  Libraries, packages, …

Test code in container image
  Container is not required to run gtest, but take effort to run on host directly
SONiC is composed of components such as each has its own build and unit test code.

ACL

- Redis (Config) – SWSS/Orchagent – Redis (ASIC) – syncd – SAI
- Swss-common/sairedis/hiredis required; SAI and Redis not necessary
- 3 levels
  - Mimic database and SAI
  - Use real database and mimic SAI
  - Use real database and virtual switch