
Tommaso Caiazzi*, Mariano Scazzariello*, Leonardo Alberro†, Lorenzo Ariemma*, Alberto Castro†, Eduardo Grampin†, Giuseppe Di Battista*

*Roma Tre University, Italy – †Universidad de la Republica, Uruguay
The Problem

Data Center networks are complex
- High density networks
- Wide variety of routing protocols with heterogeneous implementations

Bugs and errors
- Microsoft states that in its data centers: 36% of failures is caused by software bugs, and 27% is caused by configuration errors\[^1\]

Lack of standard testing procedures and tools for assessing routing protocols
- This motivates an effort in developing solutions for such a critical task

The Goal

Define a methodology for testing routing protocol implementations

Develop a framework that implements the methodology and that can be exploited to test protocol implementations
The Focus

We focus on data centers since they are the core of the modern Internet infrastructure.

We analyze fat-trees since they are the most used topologies. The framework can be simply generalized to all Clos-based topologies.

We want to verify routing protocol implementations behavior.
   We do not want to measure performance (depend on HW).
   We want to test real software implementations.
We use the RIFT draft terminology to denote a fat-tree
The Methodology

Define tests focusing on specific elements of the topology to obtain results with a general value
  E.g., Node Failure, Link Failure

Evaluate the implementations being oblivious as much as possible with respect to the actual timings
  Allows normalization of test results independently from the execution environment

Adopt a black-box method, disregarding, as much as possible, the internals of the implementations
Test Categories

Failure Tests
  Induce many types of common faults
  Measure how implementations react to such faults
  Node Failures (Leaf, Spine, ToF) and Link Failures (Leaf-Spine, Spine-ToF)

Recovery Tests
  Measure how implementations react when the network is restored
  Node Recoveries and Link Recoveries

Partitioned Fabric
  A ToF node is completely severed from access prefixes of an entire PoD by multiple link failures
Considered Aspects

For each test we analyze **four main aspects:**

Convergence
Messaging Load
Locality
Rounds
Convergence

Goal:
Check the convergence of the tested routing protocol

Approach:
After an event, it ensures that the nodes' data plane forwarding tables reach the expected state
Test is aborted after a maximum number of attempts
Messaging Load

Goal:
Determine if the protocol overhead grows smoothly

Approach:
Measure the number of packets originated by an event
Compute the packets aggregate size
Goal:
Evaluate the ability to limit the “blast radius” of an event

Approach:
Goal:
Evaluate the ability to limit the “blast radius” of an event

Approach:

Link Topological Distance

$$4 \times 1 + 16 \times 2 + 8 \times 3 = 60$$
Rounds

Goal:
Understand how packets are propagated after an event
Packets could be received all at the same time or with different timings

Approach:
Challenging to be oblivious as much as possible with respect to the timing
To do that, the methodology exploits the node-state graph
Describes how the states of nodes are synchronized by routing protocol packets
Edges represent packets exchanged by nodes. They change the state of nodes.

Each node represents a device in a certain state.

Leaf-Node Failure – BGP

(K=2, R=2)
Leaf-Node Failure – BGP (K=2, R=2)
The State-Round Value

State-Round of $<211,S_1>$ = 2
The State-Round Value

Max State-Round Value = 4
Sibyl Framework

Software framework, based on network emulation that implements the methodology

Allows to automatically deploy and run tests on virtual fat-trees

It provides all the tests of the methodology and basic templates that can be easily extended to create custom scenarios

Some utilities and callbacks are provided by the framework
  e.g., kill (restore) a device or a link
The Architecture

Controller Agent -> Worker Agent

Sibyl Agent

Agent Proxy

Sibyl Core

TestCase

Topology Generator

Network Emulator

Convergence Check

VFTGen

Kathará

Sibyl Analyzer

Analysis

Sibyl RT Calculator
How Does It Work?

During each experiment, Sibyl performs the following steps:

1. Generates and deploys a topology
2. Starts to capture all the generated PDUs on all the interfaces
3. Starts the routing protocol
4. Performs the required actions for a specific type of test
5. Performs the convergence check
6. Shutdowns the containers
7. Performs the analysis and outputs the results (can be parallelized on all the results at a later stage)
What can I do with Sibyl?
Use Case 1: Analysis of a protocol implementation (FRRRouting BGP)

**BGP** is the de-facto standard for Clos-based data centers

We use the **FRRRouting BGP** implementation
State-of-the-art configurations derived from RFC7938 and from literature

We execute all the tests of the methodology
On topologies up to **1280 nodes** and **33K interfaces**
Tests are repeated 5 times for each topology
All the results are available in the [paper](#) and in the [results' repository](#)

07/28/2022
Use Case 1: FRR BGP Results Summary

**Messaging Load:** the implementation does not manifest any performance degradation varying K and R

**Locality:** the implementation floods all the network when a failure (recovery) happens

**Rounds:** the implementation shows a very stable behaviour. Given a specific test scenario, it requires the same number of rounds to converge, the topology size does not matter
Use Case 1: FRR BGP Results Summary

Locality of a Leaf-node failure on a (K=2, R=2) fat-tree

Node-state graph of a Leaf-node failure on a (K=2, R=2) fat-tree
Sibyl can be exploited to test and evaluate PoC and prototypes.

We tested **RIFT-Python**

Only open-source available implementation of **RIFT**\(^2\) ([IETF draft](https://datatracker.ietf.org/doc/draft-ietf-rift-rift/15/))

Used by the RIFT WG to ensure the correctness of the draft

---

Use Case 2: Implementing a New Feature

RIFT-Python was missing the negative disaggregation feature

Integration pipeline built using Sibyl and the methodology:
1. Test code changes running the tests
2. Verify nodes interactions leveraging on the node-state timelines
3. Verify the expected properties of the node-state graphs
Use Case 2: Evaluating a Prototype

Comparison between RIFT-Python and FRR BGP

Only considers topologies where RIFT-Python converges
Fair since metrics do not consider temporal-tied aspects
Check the paper for all the results
Use Case 3: Testing Flooding Reduction

Recently, there has been a lot of effort in developing flooding reduction techniques for data centers\[3\][4][5][6]

Sibyl can be used to compare and evaluate flooding reduction algorithms implementations

We perform a comparison between **FRR IS-IS** and **FRR Openfabric**

You can find all the results in the repository


Spotting The Differences

IS-IS Leaf-Node Failure (K=2, R=2)  
Openfabric Leaf-Node Failure (K=2, R=2)
Other possible use cases

Detection of bug and inefficiencies using the Node-State Graph
   Look in the original paper

Protocol configurations verification

Interoperability tests

Contacts

tommaso.caiazzi@uniroma3.it
mariano.scazzariello@uniroma3.it
contact@kathara.org

Links

Sibyl Paper
Sibyl Framework Repository
Kathará Website