RIFT-Python Open Source Implementation
Status Update, Lessons Learned, and Interop Testing

Version 1, 25-Jun-2019
RIFT Working Group, IETF 105, Montreal
Author: Bruno Rijsman, brunorijsman@gmail.com
Presenter: Tony Przygienda
RIFT-Python open source implementation

- On GitHub: https://github.com/brunorijsman/rift-python
- Implemented in Python
- Intended to validate draft (reference implementation)
- Grew out of IETF 102 hackathon
- Not associated with any vendor
- The finish line is on the horizon (i.e. almost complete)
New since IETF 104

• Positive disaggregation implementation
• Flooding reduction implementation
• Security:
  • Security implementation
  • Security interop testing
  • Security review report
What is still missing in RIFT-Python?

**Plan to do:**
- Negative disaggregation
- East-west links
- Multi-neighbor state
- Processing key-value TIEs
- Policy-guided prefixes
- Setting the overload bit
- Clock comparison
- Fabric bandwidth balancing
- More test cases

**Currently no plan to do:**
- Label binding / segment routing
- BFD
- Multicast
- YANG data model

Help (GitHub pull requests) always welcome.
Positive Disaggregation
Why positive disaggregation

Spine-1-1 advertises default route but cannot reach leaf-1-1

Broken link

Traffic from leaf-1-2 or leaf-1-3 to leaf-1-1 will be blackholed if they follow the default route to spine-1-1
Positive disaggregation

Spine-1-2 and spine-1-3 do “positive disaggregation”:
(1) Detect that spine-1-1 has lost reachability to leaf-1-1
(2) Advertise specific (/32 and /128) routes for leaf-1-1

Leaf-1-2 and leaf-1-3 follow the more specific route to leaf-1-1
Positive disaggregation implementation

• No configuration needed; always enabled
• Summary of algorithm:
  • Detect blackhole
  • Trigger advertising disaggregated prefixes (more specifics)
  • Advertise disaggregated prefixes in south-TIEs
  • Install disaggregated prefixes in route table
• Show commands to understand what’s going on
Positive disaggregation blackhole detection

```
spine-1-2> show same-level-nodes

+-------------+-------------+-------------+-------------+
| Node System ID | North-bound Adjacencies | South-bound Adjacencies | Missing South-bound Adjacencies |
| 101          | 1 2 3 4     | 1002 1003    | 1001         |
| 103          | 1 2 3 4     | 1001 1002 1003 |             |
+-------------+-------------+-------------+-------------+
```
Positive disaggregation blackhole detection

```
spine-1-2> show interface veth-102a-1001b
Interface:
+-------------------------------+-------------------------------+
| Interface Name                | veth-102a-1001b              |
| Neighbor is Partially Connected | True                         |
| Nodes Causing Partial Connectivity | 101                         |
+-------------------------------+-------------------------------+
```
Positive disaggregation trigger more specifics

```
spine-1-2> show spf
...
South SPF Destinations:

+----------+-----+-------------+------+----------+
| Destination | Cost | Predecessor | Tags | Disaggregate |
+----------+-----+-------------+------+----------+
|          |     | System IDs  |      |            |
|          |     |             |      |            |
| 88.0.1.1/32 | 2   | 1001        |      | Positive   |
| 88.0.2.1/32 | 2   | 1002        |      |            |
| 88.0.3.1/32 | 2   | 1003        |      |            |
| 88.1.2.1/32 | 1   | 102         |      |            |
+----------+-----+-------------+------+----------+
```
```
leaf-1-3> show tie-db
+-----------------------------------------------+...+
| Direction | Originator | Type           |   | Contents                  |
+-----------------------------------------------+...+
| South    | 102        | Pos-Dis-Prefix |   | Pos-Dis-Prefix: 88.0.1.1/32 |
|          |            |                |   | Metric: 2                 |
+-----------------------------------------------+...+
| South    | 103        | Pos-Dis-Prefix |   | Pos-Dis-Prefix: 88.0.1.1/32 |
|          |            |                |   | Metric: 2                 |
+-----------------------------------------------+...+
```
Positive disaggregation install more specifics

```plaintext
leaf-1-3> show spf

North SPF Destinations:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost</th>
<th>Predecessor System IDs</th>
<th>IPv4 Next-hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>2</td>
<td>101</td>
<td>veth-1003a-101c 99.13.14.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>102</td>
<td>veth-1003b-102c 99.15.16.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>103</td>
<td>veth-1003c-103c 99.17.18.18</td>
</tr>
<tr>
<td>88.0.1.1/32 (Disagg)</td>
<td>3</td>
<td>102</td>
<td>veth-1003b-102c 99.15.16.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>103</td>
<td>veth-1003c-103c 99.17.18.18</td>
</tr>
</tbody>
</table>
```
**Positive disaggregation install more specifics**

```
leaf-1-3> **show route**
IPv4 Routes:
```
```
+-----------------+-------------------+-------------------+
| Prefix          | Owner             | Next-hops         |
+-----------------+-------------------+-------------------+
| 0.0.0.0/0       | North SPF         | veth-1003a-101c   |
|                 |                   | 99.13.14.14       |
|                 |                   | veth-1003b-102c   |
|                 |                   | 99.15.16.16       |
|                 |                   | veth-1003c-103c   |
|                 |                   | 99.17.18.18       |

```
```
| 88.0.1.1/32     | North SPF         | veth-1003b-102c   |
|                 |                   | 99.15.16.16       |
|                 |                   | veth-1003c-103c   |
|                 |                   | 99.17.18.18       |
+-----------------+-------------------+-------------------+
```
Flooding Reduction
Why flooding reduction?

Each super-spine node receives $N$ identical copies of each leaf TIE ($N=4$ here).
Flooding reduction: prune the flood topology

• Each leaf elects a subset of the parent spines as flood repeaters
• Make sure each grandparent super-spine receives at least R redundant copies (but typically much less than N copies)
• Try to spread the flooding burden across all spines
• $R < N$ to reduce flooding
• $R > 1$ for redundancy
Flooding reduction implementation

• Enabled by default
• Flood repeater election algorithm:
  • RIFT-Python implements example algorithm from the draft
  • Other implementations are free to choose different algorithms
  • Routers can use different algorithms and still interoperate
• Show commands to understand what’s going on
## Flooding reduction parent list

**Leaf-1-1**: show flooding-reduction

Parents:

<table>
<thead>
<tr>
<th>Interface Name</th>
<th>Parent System ID</th>
<th>Parent Interface Name</th>
<th>Grandparent Count</th>
<th>Similarity Group</th>
<th>Flood Repeater</th>
</tr>
</thead>
<tbody>
<tr>
<td>veth-1001c-103a</td>
<td>103</td>
<td>spine-1-3:veth-103a-1001c</td>
<td>4</td>
<td>1: 4-4</td>
<td>True</td>
</tr>
<tr>
<td>veth-1001b-102a</td>
<td>102</td>
<td>spine-1-2:veth-102a-1001b</td>
<td>4</td>
<td>1: 4-4</td>
<td>True</td>
</tr>
<tr>
<td>veth-1001d-104a</td>
<td>104</td>
<td>spine-1-4:veth-104a-1001d</td>
<td>4</td>
<td>1: 4-4</td>
<td>False</td>
</tr>
<tr>
<td>veth-1001a-101a</td>
<td>101</td>
<td>spine-1-1:veth-101a-1001a</td>
<td>1</td>
<td>2: 1-1</td>
<td>False</td>
</tr>
</tbody>
</table>

... continued on next slide ...

---

Provides details needed to understand the outcome of the flood repeater election algorithm.
Flooding reduction grandparent list

<table>
<thead>
<tr>
<th>Grandparent System ID</th>
<th>Parent Count</th>
<th>Flood Repeater Adjacencies</th>
<th>Redundantly Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>True</td>
</tr>
</tbody>
</table>

... continued on next slide ...

Is each grandparent redundantly covered with redundancy factor R?
# Flooding reduction interface list

<table>
<thead>
<tr>
<th>Interface Name</th>
<th>Neighbor Interface Name</th>
<th>Neighbor System ID</th>
<th>Neighbor State</th>
<th>Neighbor Direction</th>
<th>Neighbor is Flood Repeater for This Node</th>
<th>This Node is Flood Repeater for Neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>veth-1001a-101a</td>
<td>spine-1-1:veth-101a-1001a</td>
<td>101</td>
<td>THREE_WAY</td>
<td>North</td>
<td>False</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>veth-1001b-102a</td>
<td>spine-1-2:veth-102a-1001b</td>
<td>102</td>
<td>THREE_WAY</td>
<td>North</td>
<td>True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>veth-1001c-103a</td>
<td>spine-1-3:veth-103a-1001c</td>
<td>103</td>
<td>THREE_WAY</td>
<td>North</td>
<td>True</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>veth-1001d-104a</td>
<td>spine-1-4:veth-104a-1001d</td>
<td>104</td>
<td>THREE_WAY</td>
<td>North</td>
<td>False</td>
<td>Not Applicable</td>
</tr>
</tbody>
</table>

Flood repeater status per interface (both north-bound and south-bound)
Flooding reduction configuration (optional)

• Enable or disable flooding reduction
  • YAML file attribute `flooding_reduction`
  • Enabled by default

• Redundancy factor \( R \) (minimum grandparent coverage)
  • YAML file attribute `flooding_reduction_redundancy`
  • Default value 2

• Similarity factor \( S \) (to spread the flooding burden)
  • YAML file attribute `flooding_reduction_similarity`
  • Default value 2
Security
Security implementation

- Outer keys per interface
- Inner keys (aka TIE origin keys) per router
- Multiple algorithms (SHA, HMAC-SHA) and key lengths
- Support for key roll-over using optional accept keys
- Extensive statistics and logging
Security configuration example

**authentication_keys:**
- id: 1
  algorithm: sha-256
  secret: top-secret
- id: 2
  algorithm: sha-256
  secret: one-if-by-land
- id: 3
  algorithm: sha-256
  secret: two-if-by-water
- id: 4
  algorithm: hmac-sha-256
  secret: dont-tell-anyone

**nodes:**
- name: node2
  active_origin_authentication_key: 3
  accept_origin_authentication_keys: [1, 4]
- interfaces:
  - name: if1
    active_authentication_key: 1
  - name: if2
    active_authentication_key: 2
    accept_authentication_keys: [1]

Note: Some keywords will like change to agreement to change terminology from “origin” to “inner”
node1> show interface if1 security

Security Statistics:

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
<th>Last Rate Over Last 10 Changes</th>
<th>Last Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing outer security envelope</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zero outer key id not accepted</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-zero outer key id not accepted</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incorrect outer fingerprint</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missing TIE origin security envelope</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zero TIE origin key id not accepted</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-empty outer fingerprint accepted</td>
<td>109 Packets, 26138 Bytes</td>
<td>2.99 Packets/Sec, 754.24 Bytes/Sec</td>
<td>0d 00h:00m:00.77s</td>
</tr>
<tr>
<td>Non-empty origin fingerprint accepted</td>
<td>7 Packets, 1682 Bytes</td>
<td>3.04 Packets/Sec, 740.25 Bytes/Sec</td>
<td>0d 00h:00m:36.75s</td>
</tr>
<tr>
<td>Empty outer fingerprint accepted</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Empty origin fingerprint accepted</td>
<td>0 Packets, 0 Bytes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Security interop testing

• RIFT-Python – RIFT-Juniper interop testing in July 2019

• Focus on security in this round of testing
  Summary of lessons learned on next slide; details in published security review report

• Fully automated interop test suite
  Uses automated RIFT-Python test suite, but replaces one router with Juniper

• Currently all interop tests are passing
  RIFT-Python: GitHub tag ietf-105
  RIFT-Juniper: pre-release version 0.11.0-20d78c8 (Linux Customer-image)
Security lessons learned

• Outer and inner fingerprints
  • Very straightforward; got this to interoperate very quickly.
  • Agreed on new consistent terminology (e.g. inner vs origin).
  • Discussion on which fields the fingerprint should cover.

• Nonce reflection
  • Most novel part of RIFT security; quite different from OSPF and ISIS. Most lessons learned here.
  • Covers both intra-session and inter-session replay attacks (no need for storing boot-counts in non-volatile storage).
  • Must be careful to not increase nonce too aggressively.
  • Nonces have non-closable window of vulnerability of $\geq 5$ LIE intervals. But second line of defense (FSM) is quite resilient to attacks.
  • Draft was changed to use remote-nonce 0 in states 1way and 2way.
RIFT security review

• Very detailed RIFT security review report was published. Based on draft review, implementation experience, and interop testing.

• First version (very out of date now)
  Published 1-May-2019

• Second version
  Will be published soon (ETA before the end of July, will announce on mailing list)
Questions?