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

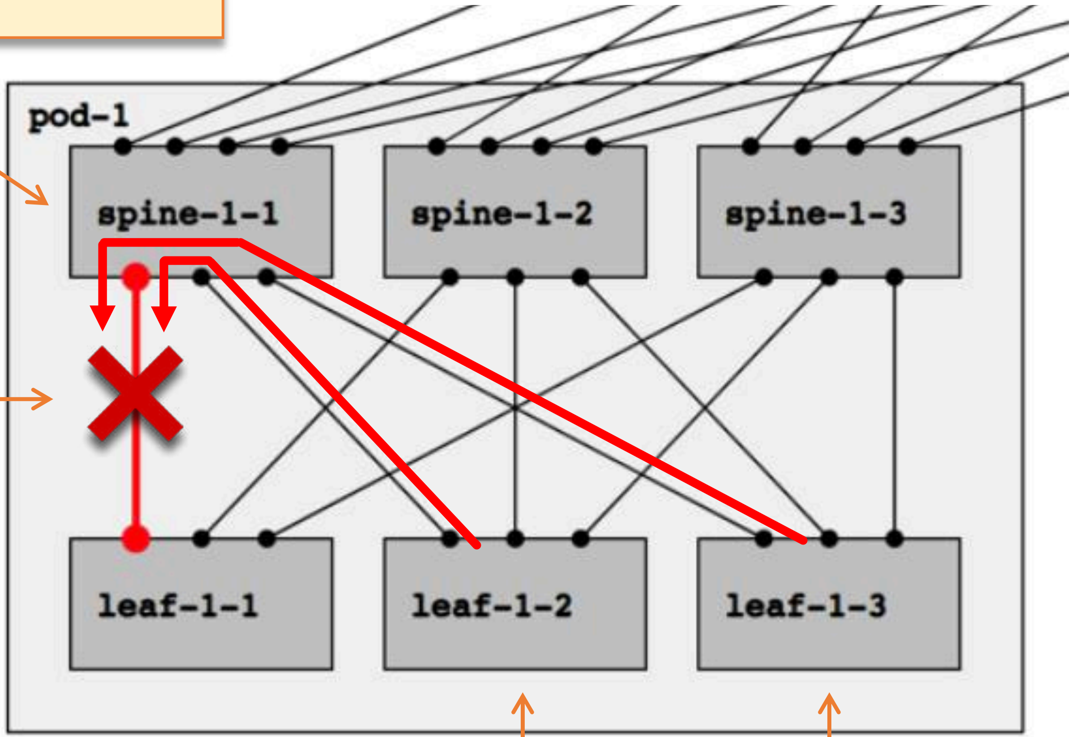
Help (GitHub pull requests) always welcome.

Positive Disaggregation

Why positive disaggregation

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

Broken link



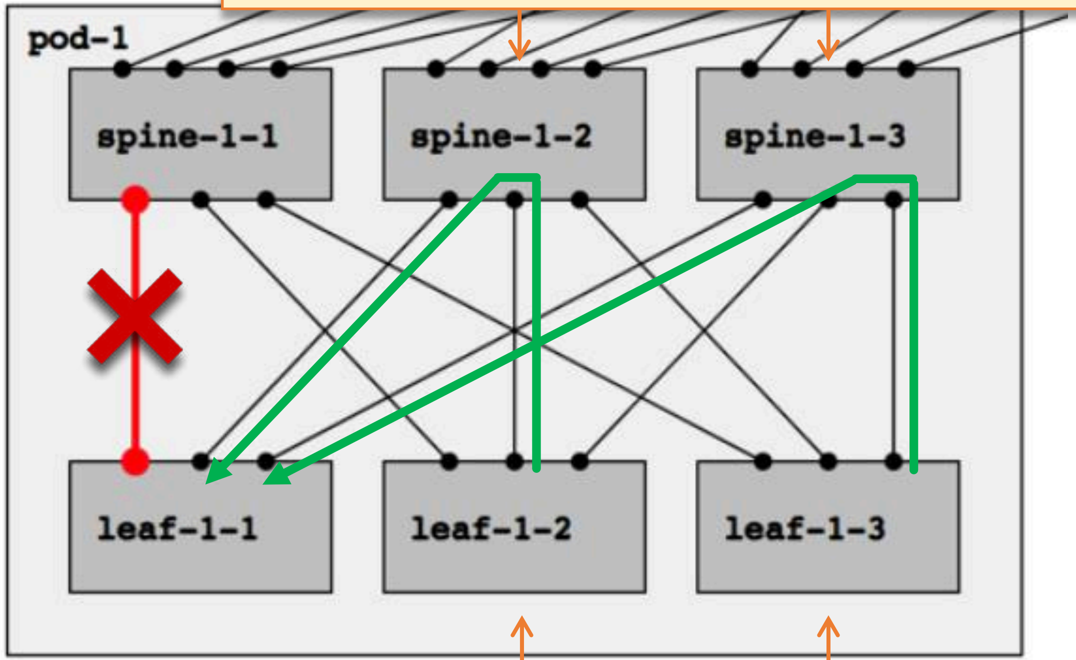
Traffic from leaf-1-2 or leaf-1-3 to leaf-1-1 will be blackholed if they follow the default route to spine-1-1

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

- Detailed feature guide: http://bit.ly/rift-python-pos-disag-feature-guide
- 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      | North-bound  | South-bound  | Missing      |
| System ID | Adjacencies  | Adjacencies  | South-bound  |
|           |              |              | Adjacencies  |
+-----------+--------------+--------------+--------------+
| 101       | 1            | 1002         | 1001         |
|           | 2            | 1003         |              |
|           | 3            |              |              |
|           | 4            |              |              |
+-----------+--------------+--------------+--------------+
| 103       | 1            | 1001         |              |
|           | 2            | 1002         |              |
|           | 3            | 1003         |              |
|           | 4            |              |              |
+-----------+--------------+--------------+--------------+
```

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          |      |              |
+-----------------+------+--------------+------+--------------+...
```

Positive disaggregation flood more specifics

```
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

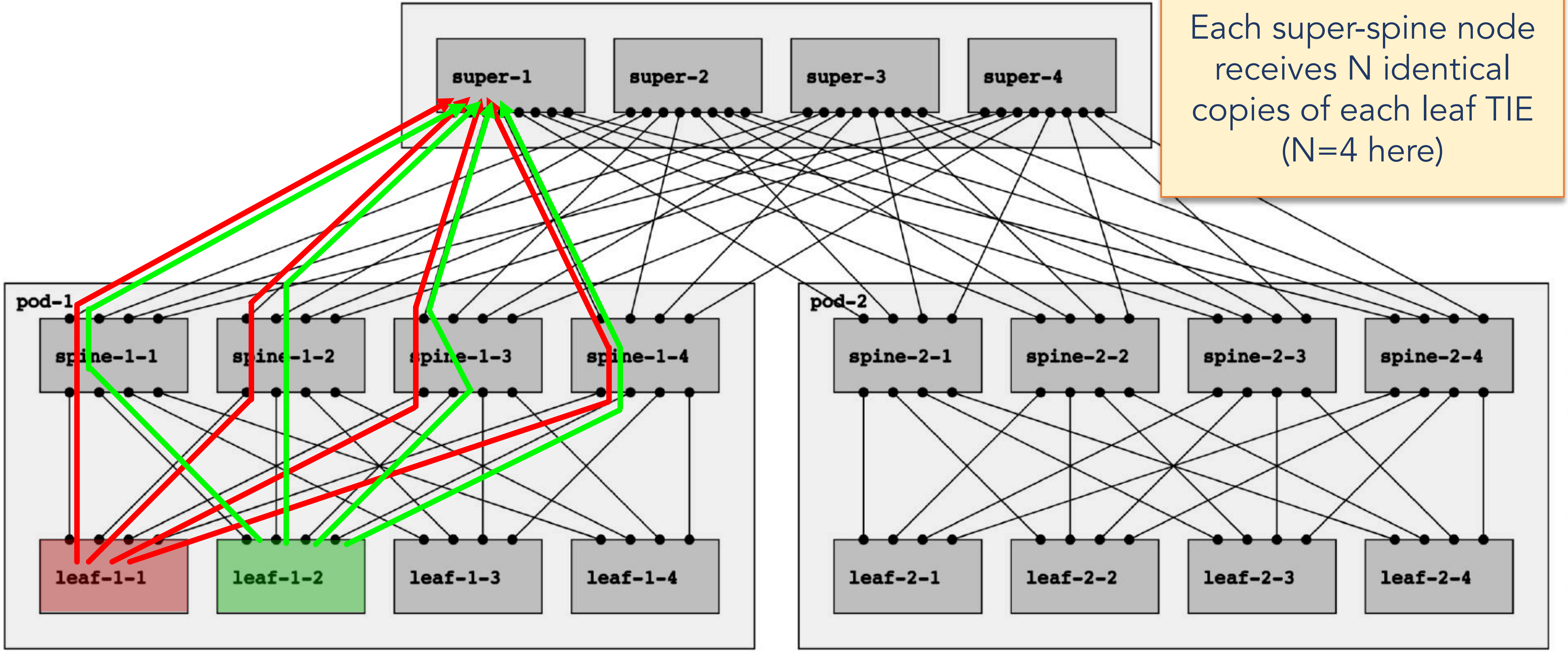
```
leaf-1-3> show spf

North SPF Destinations:
+-----------------------+------+--------------+...+------------------------------+...
| Destination           | Cost | Predecessor  |   | IPv4 Next-hops               |
|                       |      | System IDs   |   |                              |
+-----------------------+------+--------------+...+------------------------------+...
.                       .      .              .   .                              .
+-----------------------+------+--------------+...+------------------------------+...
| 0.0.0.0/0             | 2    | 101          |   | veth-1003a-101c 99.13.14.14 |
|                       |      | 102          |   | veth-1003b-102c 99.15.16.16 |
|                       |      | 103          |   | veth-1003c-103c 99.17.18.18 |
+-----------------------+------+--------------+---+------------------------------+...
.                       .      .              .   .                              .
+-----------------------+------+--------------+...+------------------------------+...
| 88.0.1.1/32 (Disagg) | 3    | 102          |   | veth-1003b-102c 99.15.16.16 |
|                       |      | 103          |   | veth-1003c-103c 99.17.18.18 |
+-----------------------+------+--------------+...+------------------------------+...
.                       .      .              .   .                              .
```

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 |
+-------------+-----------+-----------------------------+
```

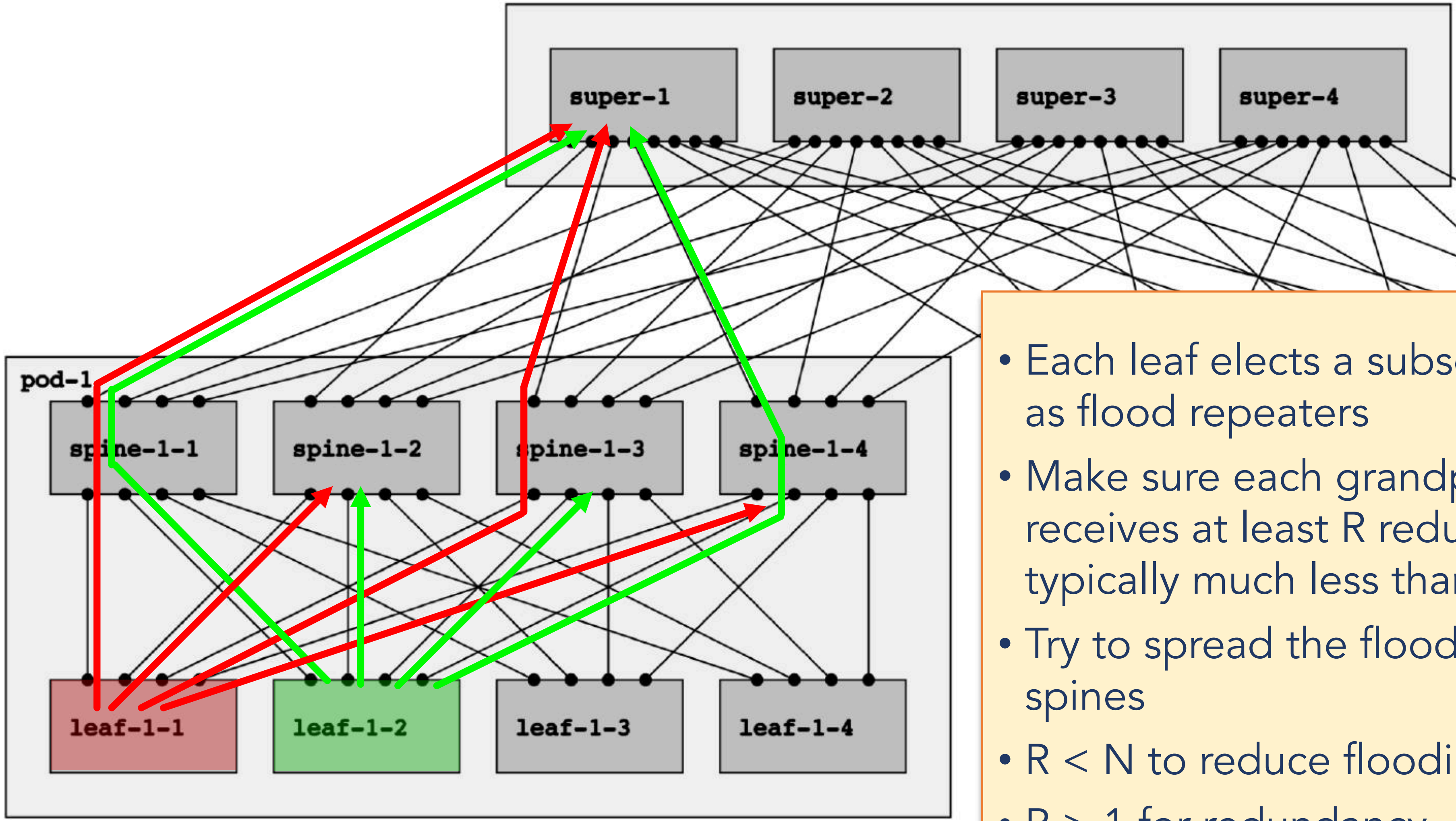
Flooding Reduction

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

- Detailed feature guide: http://bit.ly/flooding-reduction-feature-guide
- 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:
+-----------------+-----------+-------------------------+-------------+------------+----------+
| Interface       | Parent    | Parent                  | Grandparent | Similarity | Flood    |
| Name            | System ID | Interface               | Count       | Group      | Repeater |
|                 |           | Name                    |             |            |          |
+-----------------+-----------+-------------------------+-------------+------------+----------+
| veth-1001c-103a | 103       | spine-1-3:veth-103a-1001c | 4         | 1: 4-4     | True     |
+-----------------+-----------+-------------------------+-------------+------------+----------+
| veth-1001b-102a | 102       | spine-1-2:veth-102a-1001b | 4         | 1: 4-4     | True     |
+-----------------+-----------+-------------------------+-------------+------------+----------+
| veth-1001d-104a | 104       | spine-1-4:veth-104a-1001d | 4         | 1: 4-4     | False    |
+-----------------+-----------+-------------------------+-------------+------------+----------+
| veth-1001a-101a | 101       | spine-1-1:veth-101a-1001a | 1         | 2: 1-1     | False    |
+-----------------+-----------+-------------------------+-------------+------------+----------+
```

… *continued on next slide* …

Provides details needed to understand the outcome of the flood repeater election algorithm.

Flooding reduction grandparent list

Grandparents:

Grandparent System ID	Parent Count	Flood Repeater Adjacencies	Redundantly Covered
1	3	2	True
2	3	2	True
3	3	2	True
4	4	2	True

… *continued on next slide* …

Is each grandparent redundantly covered with redundancy factor R?

Flooding reduction interface list

Interfaces:

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

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

- Detailed feature guide: http://bit.ly/rift-python-security-feature-guide
- 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”

Security statistics example

```
node1> show interface if1 security
[…]
Security Statistics:
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Description                               | Value                  | Last Rate                      | Last Change      |
|                                           |                        | Over Last 10 Changes           |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Missing outer security envelope           | 0 Packets, 0 Bytes     |                                |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Zero outer key id not accepted            | 0 Packets, 0 Bytes     |                                |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Non-zero outer key id not accepted        | 0 Packets, 0 Bytes     |                                |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Incorrect outer fingerprint               | 0 Packets, 0 Bytes     |                                |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Missing TIE origin security envelope      | 0 Packets, 0 Bytes     |                                |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Zero TIE origin key id not accepted       | 0 Packets, 0 Bytes     |                                |                  |
⋮                                           ⋮                        ⋮                                ⋮                  ⋮
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Non-empty outer fingerprint accepted      | 109 Packets, 26138 Bytes | 2.99 Packets/Sec, 754.24 Bytes/Sec | 0d 00h:00m:00.77s |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Non-empty origin fingerprint accepted     | 7 Packets, 1682 Bytes  | 3.04 Packets/Sec, 740.25 Bytes/Sec | 0d 00h:00m:36.75s |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Empty outer fingerprint accepted          | 0 Packets, 0 Bytes     |                                |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
| Empty origin fingerprint accepted         | 0 Packets, 0 Bytes     |                                |                  |
+-------------------------------------------+------------------------+--------------------------------+------------------+
```

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 ≥ 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
  http://bit.ly/rift-security-review
- Second version
  Will be published soon (ETA before the end of July, will announce on mailing list)
  http://bit.ly/rift-security-review-v2

Questions?