RIFT-Python Open Source Implementation

Status Update, Lessons Learned, and Interop Testing

IETF 104, Prague, RIFT Working Group

Bruno Rijsman, brunorijsman@gmail.com, 24-Mar-2019 v2

RIFT open source implementation

- On GitHub: https://github.com/brunorijsman/rift-python
- Grew out of IETF 102 hackathon
  - Original modest goal was to test the LIE FSM
  - Work is continuing to become complete RIFT implementation
- Goals:
  - Help get the RIFT specification to the point that it is clear and complete
  - To be a reference RIFT implementation
- Current emphasis on being a user-friendly, educational, transparent, debugable reference implementation (rather than raw performance).
- Implemented in Python
- Extensive documentation: README.md
- Not associated with any vendor

Getting started with RIFT-Python

https://github.com/brunorijsman/rift-python/blob/master/README.md

build passing codecov 81%

Routing In Fat Trees (RIFT)

This repository contains a Python implementation of the Routing In Fat Trees (RIFT) protocol specified in Internet Draft (ID) draft-draft-rift-03

The code is currently still a work in progress (see Feature List below for the status).

Documentation

- Feature List
- Installation Instructions
- Startup Instructions
- Command Line Options
- Command Line Interface (CLI)
- Logging
- Log Visualization

New: IPv6 support

- IPv6 adjacencies (LIE packets)
  - Always send both IPv4 and IPv6 LIE packets (order does not matter)
  - Every LIE packet is received twice (once on IPv4 an once on IPv6)
  - LIE FSM must be (and in fact is) “idem-potent”:
  - Receiving the same LIE packet again must not change state
  - Sending an IPv6 LIE implies that IPv6 forwarding is supported (ditto for IPv4)
  - If a node wants to stop forwarding IPv4/IPv6, the adjacency must be bounced
- IPv6 flooding (TIE, TIDE, TIRE packets)
  - Pick **either** IPv4 **or** IPv6 for sending flooding packets
  - Based on first LIE packet received (this is implementation choice)
  - IPv4 routes can be flooded over IPv6 and vice versa
  - Might end up using IPv4 flooding A→B, but IPv6 flooding B→A
  - Hence, must always receive flooding packets on both IPv4 an IPv6
  - “Chaos monkey” testing found a bug here

New: IPv6 support

- IPv6 support very dependent on OS and OS version
- **`show interface … sockets`** command helps debug IPv6 issues

```
leaf-3-1> show interface veth-1007a-107a sockets
```

Traffic	Direction	Family	Local Address	Local Port	Remote Address	Remote Port
LIEs	Receive	IPv4	224.0.0.120	10000	Any	Any
LIEs	Receive	IPv6	ff02::78%veth-1007a-107a	10000	Any	Any
LIEs	Send	IPv4	99.37.38.37	41761	224.0.0.120	10000
LIEs	Send	IPv6	fe80::648e:c4ff:fed7:fcb7%veth-1007a-107a	44996	ff02::78%veth-1007a-107a	10000
Flooding	Receive	IPv4	99.37.38.37	10001	Any	Any
Flooding	Receive	IPv6	fe80::648e:c4ff:fed7:fcb7%veth-1007a-107a	10001	Any	Any
Flooding	Send	IPv4	99.37.38.37	46850	99.37.38.38	10001

New: Flooding reduction

- Implemented the "example" algorithm in draft (which is complex)
- Other implementations are free to choose different algorithm
- **`show flooding-reduction`** command for debugging:

```
leaf-3-1> show flooding-reduction
```

Parents:

Interface Name	Parent System ID	Parent Interface Name	Grandparent Count	Similarity Group	Flood Repeater
veth-1007a-107a	107	spine-3-1:veth-107a-1007a	2	1: 3-2	True
veth-1007c-109a	109	spine-3-3:veth-109a-1007c	3	1: 3-2	True
veth-1007b-108a	108	spine-3-2:veth-108a-1007b	2	1: 3-2	False

Grandparents:

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

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-1007a-107a	spine-3-1:veth-107a-1007a	107	THREE_WAY	North	True	Not Applicable
veth-1007b-108a	spine-3-2:veth-108a-1007b	108	THREE_WAY	North	False	Not Applicable
veth-1007c-109a	spine-3-3:veth-109a-1007c	109	TWO_WAY	North	True	Not Applicable

New: Shortest Path First (SPF)

- Separate north-bound and south-bound SPFs
- **`show spf`** command for debugging:

```
spine-1-1> show spf
SPF Statistics:
+---------------+------+
| SPF Runs      | 1193 |
+---------------+------+
| SPF Deferrals | 9848 |
+---------------+------+

South SPF Destinations:
+------------------+------+-------------+------+-------------------------+-----------------------------------------+
| Destination      | Cost | Predecessor | Tags | IPv4 Next-hops          | IPv6 Next-hops                          |
|                  |      | System IDs  |      |                         |                                         |
+------------------+------+-------------+------+-------------------------+-----------------------------------------+
| 101 (spine-1-1)  | 0    |             |      |                         |                                         |
+------------------+------+-------------+------+-------------------------+-----------------------------------------+
| 1002 (leaf-1-2)  | 1    | 101         |      | veth-101b-1002a 99.7.8.7 | veth-101b-1002a fe80::3477:5cff:fe85:68fd |
+------------------+------+-------------+------+-------------------------+-----------------------------------------+
| 88.0.2.1/32      | 2    | 1002        |      | veth-101b-1002a 99.7.8.7 | veth-101b-1002a fe80::3477:5cff:fe85:68fd |
+------------------+------+-------------+------+-------------------------+-----------------------------------------+
| 88.1.1.1/32      | 1    | 101         |      |                         |                                         |
+------------------+------+-------------+------+-------------------------+-----------------------------------------+

North SPF Destinations:
+------------------+------+-------------+------+----------------+----------------+
| Destination      | Cost | Predecessor | Tags | IPv4 Next-hops | IPv6 Next-hops |
|                  |      | System IDs  |      |                |                |
+------------------+------+-------------+------+----------------+----------------+
| 101 (spine-1-1)  | 0    |             |      |                |                |
+------------------+------+-------------+------+----------------+----------------+
| 88.1.1.1/32      | 1    | 101         |      |                |                |
+------------------+------+-------------+------+----------------+----------------+
```

New: Routing Information Base (RIB)

- IPv4 and IPv6
- ECMP support
- **`show routes`** commands to see contents of RIB:

```
spine-1-1> show routes
IPv4 Routes:
+-----------+-----------+-------------------------+
| Prefix    | Owner     | Next-hops               |
+-----------+-----------+-------------------------+
| 0.0.0.0/0 | North SPF | veth-101f-3a 99.91.92.91 |
+-----------+-----------+-------------------------+

IPv6 Routes:
+--------+-----------+-------------------------------------+
| Prefix | Owner     | Next-hops                           |
+--------+-----------+-------------------------------------+
| ::/0   | North SPF | veth-101e-2a fe80::5077:c3ff:fee8:1b36 |
|        |           | veth-101f-3a fe80::f8f6:86ff:fe16:742d |
|        |           | veth-101g-4a fe80::2062:a5ff:fe18:5b77 |
+--------+-----------+-------------------------------------+
```

New: Forwarding Information Base (FIB)

- Route which will be installed into kernel
- **`show forwarding`** commands to see contents of FIB:

```
spine-1-1> show forwarding
IPv4 Routes:
+-------------+-----------+-----------------------------+
| Prefix      | Owner     | Next-hops                   |
+-------------+-----------+-----------------------------+
| 0.0.0.0/0   | North SPF | veth-101f-3a 99.91.92.91    |
|             |           | veth-101g-4a 99.109.110.109 |
+-------------+-----------+-----------------------------+
| 88.0.1.1/32 | South SPF | veth-101a-1001a 99.1.2.1    |
+-------------+-----------+-----------------------------+
| 88.0.2.1/32 | South SPF | veth-101b-1002a 99.7.8.7    |
+-------------+-----------+-----------------------------+

IPv6 Routes:
+--------+-----------+---------------------------------------+
| Prefix | Owner     | Next-hops                             |
+--------+-----------+---------------------------------------+
| ::/0   | North SPF | veth-101d-1a fe80::9079:71ff:fe08:728e |
|        |           | veth-101f-3a fe80::f8f6:86ff:fe16:742d |
|        |           | veth-101g-4a fe80::2062:a5ff:fe18:5b77 |
+--------+-----------+---------------------------------------+
```

New: Install Routes into Kernel

- **`show kernel routes`** commands to see routes in kernel:

```
spine-1-1> show kernel routes table main
Kernel Routes:
```

Table	Address Family	Destination	Type	Protocol	Outgoing Interface	Gateway	Weight
Main	IPv4	99.1.2.0/24	Unicast	Kernel	veth-101a-1001a		
Main	IPv4	99.7.8.0/24	Unicast	Kernel	veth-101b-1002a		
Main	IPv4	99.13.14.0/24	Unicast	Kernel	veth-101c-1003a		
Main	IPv4	99.55.56.0/24	Unicast	Kernel	veth-101d-1a		
Main	IPv4	99.73.74.0/24	Unicast	Kernel	veth-101e-2a		
Main	IPv4	99.91.92.0/24	Unicast	Kernel	veth-101f-3a		
Main	IPv4	99.109.110.0/24	Unicast	Kernel	veth-101g-4a		
Main	IPv6	::/0	Unicast	RIFT	veth-101f-3a	fe80::f8f6:86ff:fe16:742d	
Main	IPv6	::/0	Unicast	RIFT	veth-101g-4a	fe80::2062:a5ff:fe18:5b77	
Main	IPv6	fe80::/64	Unicast	Kernel	veth-101a-1001a		
Main	IPv6	fe80::/64	Unicast	Kernel	veth-101b-1002a		

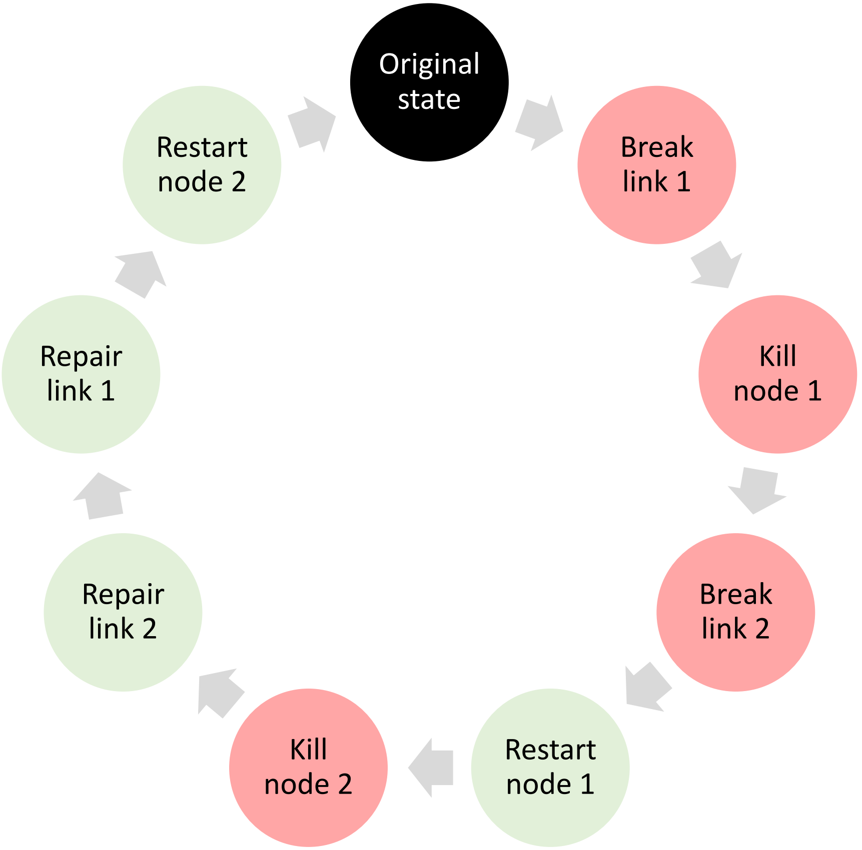
New: Extensive Statistics

- Statistics per interface, per node, per engine
- Packet count and rates, byte count and rates
- FSM events and transitions
- Show commands with exclude-zero option, clear commands

```
spine-1-1> show interface veth-101e-2a  statistics
Traffic:
+-----------------------------+------------------------------+----------------------------------------+--------------------+
| Description                 | Value                        | Last Rate                              | Last Change        |
|                             |                              | Over Last 10 Changes                   |                    |
+-----------------------------+------------------------------+----------------------------------------+--------------------+
| RX IPv4 LIE Packets         | 258 Packets, 39064 Bytes     | 0.07 Packets/Sec, 9.79 Bytes/Sec       | 0d 00h:00m:14.91s |
+-----------------------------+------------------------------+----------------------------------------+--------------------+
| TX IPv4 LIE Packets         | 5150 Packets, 792080 Bytes   | 0.75 Packets/Sec, 115.02 Bytes/Sec     | 0d 00h:00m:00.26s |
+-----------------------------+------------------------------+----------------------------------------+--------------------+
| RX IPv4 TIE Packets         | 17 Packets, 5256 Bytes       | 0.00 Packets/Sec, 1.31 Bytes/Sec       | 0d 00h:02m:28.51s |
+-----------------------------+------------------------------+----------------------------------------+--------------------+
| TX IPv4 TIE Packets         | 1074 Packets, 320971 Bytes   | 10.53 Packets/Sec, 3115.93 Bytes/Sec   | 0d 00h:00m:03.37s |
+-----------------------------+------------------------------+----------------------------------------+--------------------+
| RX IPv4 TIDE Packets        | 36 Packets, 77112 Bytes      | 0.00 Packets/Sec, 10.70 Bytes/Sec      | 0d 00h:00m:12.06s |
+-----------------------------+------------------------------+----------------------------------------+--------------------+
| TX IPv4 TIDE Packets        | 323 Packets, 252586 Bytes    | 0.09 Packets/Sec, 69.55 Bytes/Sec      | 0d 00h:00m:03.53s |
```

New: Chaos testing framework

Described in IETF 104 hackathon readout presentation



Proposal to implement RIFT in FRR

- Free Range Routing (FRR) is open source routing stack: https://frrouting.org/
- FRR community is considering implementing RIFT on FRR
- Code would be in C and open source
- The work would be done by NetDEF (a 501c3) https://www.netdef.org/
- Bruno Rijsman (author of RIFT-Python) would be main implementer
- Completely new implementation in C on FRR infrastructure, but re-using experience gained from RIFT-Python implementation
- Expose YANG data structures using new FRR north-bound APIs
- Looking for additional sponsorship. Contact Alistair Woodman (awoodman@netdef.org or in person here at IETF)if interested.

Current status summary

Feature group	Completeness estimate
Adjacencies	85%
Zero touch provisioning (ZTP)	100%
Flooding	75%
Route calculation	50%
Management interface	75%
Development toolchain	85%

- Was already completed at IETF 103
- Newly completed at IETF 104
- Not yet completed

Note: all estimates are a finger in the wind estimates

Current status: adjacencies

Complete	Not Complete
Exchange LIE packets LIE finite state machine IPv4 adjacencies Interoperability with vendor RIFT **IPv6 adjacencies**	New multi-neighbor state Interactions with BFD **Security envelope**

Current status: Zero Touch Provisioning (ZTP)

Complete	Not Complete
ZTP finite state machine Automatic level determination Interoperability with vendor RIFT	-

Current status: flooding

Complete	Not Complete
Exchange TIE / TIDE / TIRE packets	Efficient TIE propagation (w/o decode)
Node TIEs	Positive disaggregation TIEs
Prefix TIEs	Negative disaggregation TIEs
TIE database	Key-value TIEs
TX / RTX / REQ / ACK queues	External TIEs
Flooding procedures	Policy-guided prefixes
Flooding scope rules (N, S, EW)	Setting sent overload bit
South-bound default route origination	Clock comparison
Honoring received overload bit	
Interoperability with vendor RIFT	
IPv6 flooding	
Flooding reduction	

Note: large bold font indicates changes since IETF-103

Current status: route calculation

Complete	Not Complete
Routing Information Base (RIB) **Forwarding Information Base (FIB)** **North-bound SPF** **South-bound SPF** **IPv4 and IPv6** **Optimized route calculation on leafs** **ECMP**	East-west forwarding Positive disaggregation procedures Negative disaggregation procedures Fabric bandwidth balancing Label binding / segment routing **Multicast** *

* = Not yet clear whether or not this will be included in the RIFT draft

Current status: management

Complete	Partial	Not Complete
Configuration file Telnet CLI client Operational commands Documentation Multi-node topologies Logging **Command history**	On-the-fly config commands Command help **Statistics**	SSH CLI client * Command completion YANG data models * **Granular debugging / tracing**

* = Currently not planning to implement this

Current status: development toolchain

Complete	Not Complete
Automated unit tests Automated system tests Automated interop tests Travis continuous integration (CI) Strict pylint Finite state machine (FSM) framework Visualization tool **Codecov code coverage (~ 85%)** **Topology generation tool** **Network namespace-based topologies** **Chaos monkey**	100% code coverage Wireshark dissector / standalone pcap