

# RIFT Hackathon

IETF 102

# Participants

Dmitry Afanasiev	(Yandex)
Don Fedyk	(HPE)
Tony Przygienda	(Juniper)
Bruno Rijsman	(No affiliation)
Jeff Tantsura	(Nuage Networks)
Pascal Thubert	(Cisco)
Ilya Vershkov	(Mellanox)
Zhaohui (Jeffrey) Zhang	(Juniper)

# Participating RIFT Implementations

- Vendor RIFT implementation
- Open source Python RIFT implementation

# Vendor RIFT Implementation

- Contributed by Tony Przygienda
- Nearly complete implementation of draft-ietf-rift-rift-02:  
LIE FSM, ZTP FSM, Flooding FSM, IPv4, IPv6
- Binary available: <https://www.juniper.net/us/en/dm/free-rift-trial/>
- Runs on MacOS or Linux (no physical router needed)
- Can simulate large multi-node topologies defined in “topology YAML”

# Open Source Python RIFT Implementation

- Contributed by Bruno Rijsman
- Very partial implementation of draft-ietf-rift-rift-02:  
LIE FSM, IPv4
- Started implementing ZTP FSM during hackathon (not complete yet)
- Open source: <https://github.com/brunorijsman/rift-python>
- Written in Python, tested on MacOS and Linux
- Intended for experimentation, not for production
- Can simulate small multi-node topologies
- Uses same topology configuration file as vendor implementation

# Achieved LIE FSM 3-Way Adjacency in 3 hours

```
[agg_202> show interface if_202_1
```

Interface:

Interface Name	if_202_1
Advertised Name	agg_202-if_202_1
Interface IPv4 Address	
Metric	1
Receive LIE IPv4 Multicast Address	224.0.100.2
Transmit LIE IPv4 Multicast Address	224.0.100.1
Receive LIE IPv6 Multicast Address	FF02::0078
Transmit LIE IPv6 Multicast Address	FF02::0078
Receive LIE Port	20021
Transmit LIE Port	20022
Receive TIE Port	20024
System ID	202
Local ID	1
MTU	1500
POD	0
State	THREE_WAY
Neighbor	yes

Neighbor:

Name	None
System ID	1
IPv4 Address	172.31.16.255
LIE UDP Source Port	20021
Link ID	28672
Level	2
Flood UDP Port	20023
MTU	1400
POD	0
Hold Time	3
Not a ZTP Offer	False
You Are Not a ZTP Flood Repeater	False
Your System ID	202
Your Local ID	1

```
Jul 18 20:16:39.243 DEBG received reflection first time, rebuild packet
```

```
Jul 18 20:16:39.244 DEBG adjacency 3-way up
```

# Summary of Results

- Quickly achieved interoperability between vendor and Python RIFT:  
**LIE FSM adjacency in state 3-way (IPv4)**
- Very detailed interoperability report:  
<http://bit.ly/ietf-102-rift-hackathon-interop-report>
- Started implementation of ZTP FSM in Python RIFT  
Expected to be completed in weeks
- Detailed review of ietf-draft-rift-rift-02:  
<http://bit.ly/rift-comments>  
Live document for other commenters and author responses
- Additional minor comments on draft while implementing ZTP

# Summary of Lessons Learned

- Model-based protocol encoding works and has great benefits
- Multicast is very platform dependent
- Attending a hackathon is very useful even for non-coders



# Model-Based Protocol Encoding

- RIFT uses Thrift to model protocol packet encoding
- All packet encoding and decoding code is generated
  - Can implement encoding and decoding in a matter of minutes
  - High confidence in correctness of encoding and decoding code
- Discovered limitations of Thrift
  - Thrift does not support unsigned integers (only signed i8, i16, i32, i64)
    - The RIFT draft specifies that certain signed integers MUST be treated as unsigned
    - This severely dilutes the value of code auto-generation (must manually “fudge” fields)
    - Actually caused platform-dependent crash (encoded IP address out of range in Python)
  - Existing Thrift framework for transport cannot be used (UDP is not supported)
- **Bottom line: using model-based encoding works and helps a lot**

# Multicast is Very Platform-Dependent

- Particularly IPv6 multicasts
- Different socket options across platforms
- Different behavior of same socket options across platforms

# Follow-up Work

- Finish Python RIFT
  - ZTP FSM, flooding FSM, IPv6, ...
  - Considering port to Free Range Router (FRR)
- More interoperability testing
  - In future IETF hackathons
  - In interim hackathon (ZTP state machine)
- Convergence and correctness testing in addition to interop testing
  - Performance: How quickly does RIFT (re)converge
  - Correctness: Does RIFT (re)converge to the correct tables in all scenarios
- More RIFT implementations are welcome to join