Lost in Fat Tree forest and route out

RIFT: Novel DC Fabric Routing Protocol (draft-przygienda-rift)

Rafal Szarecki
Solution Architect; Juniper Networks
Content

- Blitz overview of today’s routing
- DC fabric routing is a specialized problem
- RIFT: a novel routing algorithm for CLOS underlay
Blitz Overview of Today’s Routing

• Link-State & SPF
• Distance/Path Vector
Link State and SPF = Distributed Computation

- Topology elements - nodes, links, prefixes
- Each node originates packets with its elements
- Packets are "flooded"
- "Newest" version wins
- Each node "sees" whole topology
- Each node "computes" reachability to everywhere
- Conversion is very fast
- Every link failure shakes whole network
- Flooding generates excessive load for large average connectivity
- Periodic refreshes

Examples: OSPF, IS-IS, PNNI, TRILL, RBridges
Distance/Path Vector = Diffused Computation

- Prefixes “gather” metric when passed along links
- Each sink computes “best” result and passes it on (Add-Path changed that)
- A “sink” keeps all copies, otherwise it would have to trigger “re-diffusion”
- Loop prevention is easy on strictly uniformly increasing metric.
- Ideal for “policy” rather than “reachability”
- Scales when properly implemented to much higher # of routes than Link-State

Examples: BGP, RIP, IGRP
Link State vs Distance Vector

- **Link State**
  - Topology view \(\rightarrow\) TE enabler

- **Distance/Path Vector**
  - Every computation could enforce policy – granular control – TE

- Both - Current implementation for any-topology.
DC Fabric Routing: a Specialized Problem

- Clos and Fat-Tree topologies
- Current state of dynamic DC routing
- Dynamic DC routing requirements matrix
Clos Topologies

- Clos offers well-understood blocking probabilities
- Work done at AT&T (Bell Systems) in 1950s for crossbar scaling
- Fully connected CLOS is dense and expensive
- Data centers today tend to be variations of “folded Fat-Tree”:
  - Input stages are same as output Stages
  - CLOS w/ \( m \geq n \)
Current State of Affairs

- Several of large DC fabrics use E-BGP with band-aids as IGP (RFC7938)
  - "looping paths" (allow-as)
  - “Relaxed Multi-Path ECMP”
  - AS numbering schemes to control “path hunting” via policies
  - AddPaths to support multi-homing, ECMP on EBGP
  - Efforts to get around 65K ASes and limited private AS space
  - Proprietary provisioning and configuration solutions, LLDP Extensions
  - “Violations” of FSM like restart timers and minimum-route-advertisement timers
- Others run IGP (ISIS)
- Yet others run BGP over IGP (traditional routing architecture)
- Less than more successful attempts @ prefix summarization, micro- and black-Holing
  - Works better for single-tenant fabrics without LAN stretch or VM mobility
## Dynamic DC Routing Requirements Breakdown (RFC7938+)

<table>
<thead>
<tr>
<th>Problem / Attempted Solution</th>
<th>BGP modified for DC (all kind of “mods”)</th>
<th>ISIS modified for DC (RFC7356 + “mods”)</th>
<th>RIFT Native DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Discovery/Automatic Forming of Trees/Preventing Cabling Violations</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Minimal Amount of Routes/Information on ToRs</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>High Degree of ECMP (BGP needs lots knobs, memory, own-AS-path violations) and ideally NEC and LFA</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Traffic Engineering by Next-Hops, Prefix Modifications</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>See All Links in Topology to Support PCE/SR</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Carry Opaque Configuration Data (Key-Value) Efficiently</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Take a Node out of Production Quickly and Without Disruption</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Automatic Disaggregation on Failures to Prevent Black-Holing and Back-Hauling</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Minimal Blast Radius on Failures (On Failure Smallest Possible Part of the Network “Shakes”)</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fastest Possible Convergence on Failures</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Simplest Initial Implementation</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Summary of RIFT Advantages

- Advantages of Link-State and Distance Vector
  - Fastest possible convergence
  - Automatic detection of topology
  - Minimal routes on TORs
  - High degree of ECMP
  - Fast De-commissioning of Nodes

- No disadvantages of Link-State or Distance Vector
  - Reduced flooding
  - Automatic neighbor detection

- Only RIFT can do
  - Automatic disaggregation on failures
  - Minimal blast radius on failures
  - Key-Value Store
RIFT: Novel Dynamic Routing Algorithm for Clos Underlay

• General concept
• Automatic cabling constraints
• Automatic disaggregation on failures
• Automatic flooding reduction
• Other

“Just because the standard provides a cliff in front of you, you are not necessarily required to jump off it.”
— Norman Diamond
In One Picture: Link-State Up, Distance Vector Down & Bounce

- Link-State flood Up (North)
  - Full topology and all pfx @ top spine only.

- Distance Vector down.
  - 0/0 is sufficient to send traffic UP.
  - More specific prefixes
    - disaggregated in case of failure.
    - TE

- Flood reduction and automatic dis-aggregation
Adjacency Formation

- Link Information Element
  - POD #
  - Level #
  - Node ID
- Transported over well known m-cast address and port
- POD # == 0 “Any POD”
  - Node derive POD from 1st Northbound neighbor it establish adjacency.
  - Auto-configuration
- Level # == 0 “Leaf”
Automatic Topology Constraints

Automatic rejection of adjacencies based on minimum configuration

- A1 to B1 forbidden due to POD mismatch
- A0 to B1 forbidden due to POD mismatch (A0 already formed A0-A1 even if POD not configured on A0)
- B0 to C0 forbidden based on level mismatch
Topography Information Element

- TIE processed differently when
  - Sent NorthBound – N-TIE – Link-State like
  - Send SouthBand – S-TIE – Distance-Vector like

- TIE Types
  - Node TIE – similar to ISIS LSP
  - Prefix TIE – similar to ISIS IP reachability TLV
  - PGPrefix TIE – similar to BGP NLRI
  - KeyValue TIE -
## Topology Information Element

<table>
<thead>
<tr>
<th>Node-TIE</th>
<th>Prefix-TIE</th>
<th>PGP-TIE</th>
<th>KV-TIE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content &amp; Purpose</strong></td>
<td>Node-ID, neighbors and links. Topology information.</td>
<td>IP prefixes w/ metrics</td>
<td>TE</td>
</tr>
<tr>
<td><strong>North-TIE Processing (Rx on South IF)</strong></td>
<td>Flood on all North Bound IF w/o change. Build LSDB for south bound part of fabric. Calculate SPF.</td>
<td>Flood on all North Bound IF w/o change. Build LSDB for south bound part of fabric. Calculate SPF. [Similar to ISIS’s IP reachability TLV]</td>
<td>---</td>
</tr>
<tr>
<td><strong>South-TIE Processing (Rx on North IF)</strong></td>
<td>Reflect/bounce back to all North Bound IF. Discover “Equally Connected Group”</td>
<td>Reflect/bounce back to all North Bound IF. Consume, and populate RIB Generate new on all South-Band IF – 0/0 always. More specific if needed. [Similar to aggregate route in BGP or Summary LSA]</td>
<td>---</td>
</tr>
</tbody>
</table>
Routing in steady state – basics (1)

• **Leafs**
  • Only 0/0 to connected level 1 spines.

• **Spine 111 [112]**
  • 0/0 to S31, S32 [S33, S34]
  • Pfx A to L101
  • Pfx B to L102

• **Spine 211 [212]**
  • 0/0 to S31, S32 [S33, S34]
  • Pfx C to L201
  • Pfx D to L202

• **Spine 31, 32, 33, 34**
  • Pfx A to S111, S112
  • Pfx B to S111, S112
  • Pfx C to S211, S212
  • Pfx D to S211, S212

**TIE(x)** – TIE originated at node X.
S-TIE reflection
“Equal connectivity group” discovery

1) Spine @ level X [S31] sent S-TIE to node @ level (X-1) [S111]
2) Node @ level (X-1) [S111] send S-TIE up to all neighbors [S32]
3) Spine that received bounced S-TIE [S32] compares their neighbors w/ one in S-TIE
4) Discovered “Equal connectivity group”
   1) Disaggregation
   2) Flood reduction
Routing in failure – automatic disaggregation

1) Spine X [S32] receive bounced S-TIE(31)

2) Discovery
   - Neighbor not matches – one [S211] is missing in S-TIE(S31)
   - Spine Y [S31] has no connectivity to some pfx (pfx: C, D).
   - As node in lower level (Level 1) use 0/0 – risk of black hole/losses.

3) Spine X [S32] originate new S-TIE(32) w/ disaggregated prefixes (C,D)

Note:
   Nodes on lower level (Level 1) get more specific route.
   Nodes further down [L101, L102] still can use 0/0 only
Highly mesh topology

- N-port spine switch
- Level 2 spine – all N ports are southbound
- Level 1 spine
  - N/2 ports are Southbound
  - N/2 ports are Northbound
- Link-State Flooding become over-kill
Flooding w/o Reduction

- A lot of redundant information
- Known problem in Link-State protocols in Highly meshed networks
Flooding w/o Reduction

- Each “B” node computes from reflected south representation of other “B” nodes
  - Set of South neighbors
  - Set of North neighbors
- Nodes having both sets matching consider themselves “Flood Reduction Group” and load-balance flooding
- Fully distributed, unsynchronized election
  - In this example case B1 & B2
  - Each node chooses based on hash computation which other nodes’ Information it forwards on first flood attempt
- Similar to DF election in EVPN
Moreover

- Traffic engineering is included via “flooded distance vector overlay” including filtering policies like BGP
- Packet formats are completely model based
- Channel agnostic delivery, could be QUIC, TCP, UDP, etc
- Prefixes are mapped to flooding element based on local hash functions
  - One extreme point is a prefix per flooded element = BGP update
- Purging (given complexity) is omitted
- Key-Value Store is supported (e.g. service configuration during flooding)
STATUS

• Standardization
  • Individual contribution to IETF Routing WG
  • Base for further work toward I-D

• Implementation
  • Prototype reference code exist
  • PoC Test runs, performance data collected

• Cooperation
  • Join work at IETF WG
  • Contact authors, share opinion
  • The data structures for packet are public (GPB) – draft.
Thank you
Automatic De-Aggregation

- South Representation of the Red Spine is Reflected by the Green Layer
- Lower Red Spine Switch Sees that Upper Node has No Adjacency to the Only available Next-Hop to P1
- Lower Red Node Disaggregates P1
Flooding w/o Reduction

- Not CLOS topology, but Fat-Tree
- A lot of redundant information
Flooding Reduction

- Not CLOS topology, but Fat-Tree
- Members of ECG
  - runs same Hash on SystemID of N-TIE.
  - Decide which N-TIE would be flooded
    Nort by which ECG member
Automatic Flooding Reduction

- Each “B” Node Computes From Reflected South Representation of Other “B” Nodes
  - Set of South Neighbors
  - Set of North Neighbors
- Nodes Having Both Sets Matching Consider Themselves “Flood Reduction Group” and Load-Balance Flooding
- Fully Distributed, Unsynchronized Election
- In this Example Case B1 & B2
- Each Node Chooses Based on Hash Computation which Other Nodes’ Information it Forwards on First Flood Attempt
- Similar to DF Election in EVPN