Flexible Algorithms

Shraddha Hegde
Feb 2021
Nanog 81
What is Flex-algo?

• IGP traditionally computes best effort path
  – Based on IGP metric

• Flex-algo provides a way to compute TE paths in IGP
  – Based on various constraints
  – TE metric, latency metric
  – Admin color constraints
  – Avoid node constraints

• Backup paths also honor constraints

• Being standardized in IETF LSR WG draft-ietf-lsr-flex-algo
Why Flex-algo?

• Requirements
  – Strict TE constraints
    • Avoid nodes/links
    • Avoid traffic going in another plane
  – Honor the constraints for backup paths
    • TI-LFA backup paths to honor constraints

• Possible Alternates
  – SR-TE based solution
    • Compressed label stacks having Node-SIDs may not honor constraints during convergence
    • TI-LFA backup paths do not honor constraints

Flex-algo uses single label, satisfies strict TE constraints.
Use cases

• Routing plane separation
  – Multiple routing planes with strict plane separation requirements

• Data Sovereignty
  – Strictly avoid nodes and links in certain geographical locations

• Merging two networks into one
  – Yet maintain the isolation for certain traffic

• Low latency routing
Use cases

- **Routing plane separation**
  - Strict traffic isolation between Red and Blue plane
  - If a plane is partitioned, traffic should drop and never switch to another plane
• Data Sovereignty
  – Strictly avoid nodes and links in certain geographical locations
Low latency / high bandwidth paths
Policy

• All flows follow the lowest latency path available
  • In this network, latency is a function of circuit length
• However, high bandwidth flows must avoid 10G links
# Link Advertisements

<table>
<thead>
<tr>
<th>Link</th>
<th>IGP Metric</th>
<th>TE Metric</th>
<th>Administrative Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1-R2</td>
<td>400</td>
<td>400</td>
<td>Blue</td>
</tr>
<tr>
<td>R1-R3</td>
<td>400</td>
<td>400</td>
<td>Blue</td>
</tr>
<tr>
<td>R1-R4</td>
<td>300</td>
<td>300</td>
<td>Red</td>
</tr>
<tr>
<td>R2-R4</td>
<td>400</td>
<td>400</td>
<td>Blue</td>
</tr>
<tr>
<td>R3-R4</td>
<td>400</td>
<td>400</td>
<td>Blue</td>
</tr>
</tbody>
</table>
## Flex-algo Definitions (FAD)

<table>
<thead>
<tr>
<th>FAD</th>
<th>Metric Type</th>
<th>Calculation Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Latency</td>
<td>IGP</td>
<td>SPF</td>
<td>Include all</td>
</tr>
<tr>
<td>High bandwidth</td>
<td>IGP</td>
<td>SPF</td>
<td>Exclude red</td>
</tr>
</tbody>
</table>
Pulling it together

• R4 advertises Segment A
  - Associates it with the low latency FAD

• R4 advertises Segment B
  - Associates it with the high bandwidth FAD

• R1 calculates the least-cost path to Segment A
  - Next Hop is R4
  - Because low latency FAD includes all links

• R1 calculates the least-cost path to Segment B
  - Next Hop is ECMP (either R2 or R3)
  - Because high bandwidth FAD excludes red links
Path Diversity
Example Constraints

- Red flows traverse red links
  - And no others
- Orange flows prefer red links
  - But can fail over to blue links
- Blue flows traverse blue links
  - And no others
- Yellow flows prefer blue links
  - But can fail over to red links
## Link Advertisements

<table>
<thead>
<tr>
<th>Link</th>
<th>IGP Metric</th>
<th>TE Metric</th>
<th>Administrative Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1-R2</td>
<td>200</td>
<td>400</td>
<td>Red</td>
</tr>
<tr>
<td>R1-R3</td>
<td>400</td>
<td>200</td>
<td>Blue</td>
</tr>
<tr>
<td>R2-R4</td>
<td>200</td>
<td>400</td>
<td>Red</td>
</tr>
<tr>
<td>R3-R4</td>
<td>400</td>
<td>200</td>
<td>Blue</td>
</tr>
</tbody>
</table>
# Flex-algo Definitions (FAD)

<table>
<thead>
<tr>
<th>FAD</th>
<th>Metric Type</th>
<th>Calculation Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>IGP</td>
<td>SPF</td>
<td>Exclude blue</td>
</tr>
<tr>
<td>Orange</td>
<td>IGP</td>
<td>SPF</td>
<td>Include all</td>
</tr>
<tr>
<td>Blue</td>
<td>TE</td>
<td>SPF</td>
<td>Exclude red</td>
</tr>
<tr>
<td>Yellow</td>
<td>TE</td>
<td>SPF</td>
<td>Include all</td>
</tr>
</tbody>
</table>
Pulling it together

• R4 advertises four prefix segments
  – Segment A associated with the red FAD
  – Segment B associated with the orange FAD
  – Segment C associated with the blue FAD
  – Segment D associated with the yellow FAD

• R1 calculates the least-cost path to R4 four times
  – Once for each FAD / prefix segment
R1 Routes to R4

• Via Prefix A (red)
  - Next Hop is R2
  - No failover, because red FAD excludes blue links

• Via Prefix B (orange)
  - Next Hop is R2
    • Because orange FAD uses IGP metrics
    • Because IGP metrics are lower on red links
  - Failover is R3, because orange FAD includes all links
R1 Routes to R4 (Continued)

• Via Prefix Segment C (blue)
  - Next Hop is R3
  - No failover, because blue FAD excludes red links

• Via Prefix Segment D (yellow)
  - Next Hop is R3
    • Because yellow FAD uses TE metrics
    • Because TE metrics are lower on blue links
  - Failover is R2, because yellow FAD includes all links
IP Flex-algo

• Plain IPv4/IPv6 Network
  • No MPLS! No SRv6!

• Multiple Loopbacks
  • Associate each loopback with a Flex-algo
    • Reuse FAD procedures for draft-lsr-flex-algo
    • Reuse computation procedures from draft-lsr-flex-algo
  • Loopbacks corresponding to Flex-algo follow specific path
    • Next-hops for each loopback computed based on that flex-algo
  • Service prefixes carry different loopbacks as protocol next-hops
    • Ip-in-IP tunneling used to carry services
  • Being standardized in IETF draft-ietf-lsr-ip-flexalgo
Design Guidelines

• How many Flex-algos?
  - Total number of Flex-algos a router needs to participate should be in the order 2-16
  - 100s of flex-algos in the network is not advisable

• How often a Flex-algo definition needs to change?
  - The FAD definitions should not change very often and should be stable
    • Ex: Flex-algo 128 based on delay metric
    • Flex-algo 129 based on TE metric etc.
    • Flex-algo 130 exclude red links
      The FAD definition for flex-algo 130 will not change whereas the link colors can change more often
Design Guidelines

• Migrations
  - Every node that needs to be part of flex-algo need to support Flex-algo extensions
  - Legacy nodes that do not support Flex-algo to be not included as part of Flex-algo
  - Co-existence with LDP/RSVP
Flex-Algo Operational Requirements

- MPLS ping and traceroute on flex-algo labels
  - Control plane/Data plane synchronization and validation with MPLS ping and traceroute
- Ability to count traffic per Flex-algo SID
  - Build traffic matrix on a per Flex-algo basis
- Display of Flex-algo definitions, participation details, FAD winner, details of winning FAD
  - Debugging problems in Flex-algo participation
Flex-Algo Operational Requirements

- Display of Flex-algo topology
  - Debugging problems in topology derivation
- Display of Flex-algo SPF log details with reasons
  - Debugging problems with SPFs
- Display of Flex-algo routes
  - Debugging problems in downloading Flex-algo routes
Flex-algo is powerful

- Many networks require only course-grained TE
  - As in the use-cases described above
- Benefits of deploying Flex-algo into such networks
  - Each SR path is reduced to a single segment
  - No need to specify TE policy on a controller or on each segment egress node
  - Operational simplicity
- Benefits of deploying IP Flex-algo
  - No MPLS Required
  - No large address blocks per Flex-Algo required
  - No new protocols required, uses only IGP
References

- draft-ietf-lsr-flex-algo
- draft-ietf-lsr-ip-flex-algo
Thank you

Feb 2021