RIFT
Internet-Draft
Intended status: Standards Track
Expires: October 27, 2019

A. Atlas
Individual
Z. Zhang
Juniper Networks

April 25, 2019

Policy Guided Prefixes with Routing In Fat Trees
draft-atlas-rift-pgp-01

Abstract

In a fat tree, it can be sometimes desirable to guide traffic to particular destinations or keep specific flows to certain paths. In RIFT, this traffic steering/engineering is done by using policy-guided prefixes with their associated communities. Routes based on policy-guided prefixes are preferred over regular routes. Any node can originate a policy-guided prefix and advertise it in both north and south directions, and the calculation in both directions are distance vector based.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 27, 2019.

Table of Contents

1. Introduction

In a fat tree, it can be sometimes desirable to guide traffic to particular destinations or keep specific flows to certain paths. In RIFT, this is done by using policy-guided prefixes with their associated communities. Each community is an abstract value whose meaning is determined by configuration. It is assumed that the fabric is under a single administrative control so that the meaning and intent of the communities is understood by all the nodes in the fabric. Any node can originate a policy-guided prefix.

Since RIFT uses distance vector concepts in a southbound direction, it is straightforward to add a policy-guided prefix to an S-TIE. For easier troubleshooting, the approach taken in RIFT is that a node's southbound policy-guided prefixes are sent in its S-TIE and the receiver does inbound filtering based on the associated communities (an egress policy is imaginable but would lead to different S-TIEs per adjacency possibly which is not considered in RIFT protocol

procedures). A southbound policy-guided prefix can only use links in the south direction. If an PGP S-TIE is received on an East-West or northbound link, it must be discarded by ingress filtering.

Conceptually, a southbound policy-guided prefix guides traffic from the leaves up to at most the north-most level. It is also necessary to to have northbound policy-guided prefixes to guide traffic from the north-most level down to the appropriate leaves. Therefore, RIFT includes northbound policy-guided prefixes in its N PGP-TIE and the receiver does inbound filtering based on the associated communities. A northbound policy-guided prefix can only use links in the northern direction. If an N PGP TIE is received on an East-West or southbound link, it must be discarded by ingress filtering.

By separating southbound and northbound policy-guided prefixes and requiring that the cost associated with a PGP is strictly monotonically increasing at each hop, the path cannot loop. Because the costs are strictly increasing, it is not possible to have a loop between a northbound PGP and a southbound PGP. If East-West links were to be allowed, then looping could occur and issues such as counting to infinity would become an issue to be solved (if complete generality of path - such as including East-West links and using both north and south links in arbitrary sequence - then a Path Vector protocol or a similar solution must be considered).

Besides the usage for traffic engineering, PGPs can also be used to ensure nodes are administratively reachable for debugging purpose after certain failures. For example, a node looses all its northbound adjacencies but is not at the top of the fabric. If it detects that some other members at its level are advertising northbound adjacencies MAY inject its loopback address into southbound PGP TIE and become reachable "from the south" that way. Further, a solution may be implemented where based on e.g. a "well known" community such a southbound PGP is reflected at level 0 and advertised as northbound PGP again to allow for "reachability from the north" at the cost of additional flooding.

2. Specification

PGPs are advertised in PGPrefixTIEs included in PGP N/S-TIEs. S-PGPs are propagated in south direction only and N-PGPs follow northern direction strictly. THRIFT schema in the base RIFT specification needs to be updated. For example:

- TIEElement needs to add "7: optional PGPrefixElement pog_prefixes;"

- "struct PGPrefixElement" needs to be defined. Should PrefixAttributes be used for PGPrefixElement (do all defined fields in PrefixAttributes apply to PGPrefixElement)?

- "struct Community" needs to be referenced in PGPrefixElement

Future revisions of this document and the base RIFT specification will coordinate the THRIFT schema.

2.1. Ingress Filtering

The set of policy-guided prefixes received in a TIE is subject to ingress filtering and then re-originated to be sent out in the receiver's appropriate TIE. Both the ingress filtering and the re-origination use the communities associated with the policy-guided prefixes to determine the correct behavior. The cost on re-advertisement MUST increase in a strictly monotonic fashion.

When a node X receives a PGP S-TIE or a PGP N-TIE that is originated from a node Y which does not have an adjacency with X, all PGPs in such a TIE MUST be filtered. Similarly, if node Y is at the same level as node X, then X MUST filter out PGPs in such S- and N-TIEs to prevent loops.

Next, policy can be applied to determine which policy-guided prefixes to accept. Since ingress filtering is chosen rather than egress filtering and per-neighbor PGPs, policy that applies to links is done at the receiver. Because the RIFT adjacency is between nodes and there may be parallel links between the two nodes, the policy-guided prefix is considered to start with the next-hop set that has all links to the originating node Y.

A policy-guided prefix has or is assigned the following attributes:

cost: This is initialized to the cost received

community_list: This is initialized to the list of the communities received.

next_hop_set: This is initialized to the set of links to the originating node Y.

2.2. Applying Policy

The specific action to apply based upon a community is deployment specific. Here are some examples of things that can be done with communities. The length of a community is a 64 bits number and it can be written as a single field M or as a multi-field (S = M[0-31],

T = M[32-63]) in these examples. For simplicity, the policy-guided prefix is referred to as P, the processing node as X and the originator as Y.

Prune Next-Hops: Community Required: For each next-hop in P.next_hop_set, if the next-hop does not have the community, prune that next-hop from P.next_hop_set.

Prune Next-Hops: Avoid Community: For each next-hop in P.next_hop_set, if the next-hop has the community, prune that next-hop from P.next_hop_set.

Drop if Community: If node X has community M, discard P.

Drop if not Community: If node X does not have the community M, discard P.

Prune to ifIndex T: For each next-hop in P.next_hop_set, if the next-hop's ifIndex is not the value T specified in the community (S,T), then prune that next-hop from P.next_hop_set.

Add Cost T: For each appearance of community S in P.community_list, if the node X has community S, then add T to P.cost.

Accumulate Min-BW T: Let bw be the sum of the bandwidth for P.next_hop_set. If that sum is less than T, then replace (S,T) with (S, bw).

Add Community T if Node matches S: If the node X has community S, then add community T to P.community_list.

2.3. Store Policy-Guided Prefix for Route Computation and Regeneration

Once a policy-guided prefix has completed ingress filtering and policy, it is almost ready to store and use. It is still necessary to adjust the cost of the prefix to account for the link from the computing node X to the originating neighbor node Y.

There are three different policies that can be used:

Minimum Equal-Cost: Find the lowest cost C next-hops in P.next_hop_set and prune to those. Add C to P.cost.

Minimum Unequal-Cost: Find the lowest cost C next-hop in P.next_hop_set. Add C to P.cost.

Maximum Unequal-Cost: Find the highest cost C next-hop in P.next_hop_set. Add C to P.cost.

The default policy is Minimum Unequal-Cost but well-known communities can be defined to get the other behaviors.

Regardless of the policy used, a node MUST store a PGP cost that is at least 1 greater than the PGP cost received. This enforces the strictly monotonically increasing condition that avoids loops.

Two databases of PGPs - from N-TIEs and from S-TIEs are stored. When a PGP is inserted into the appropriate database, the usual tie-breaking on cost is performed. Observe that the node retains all PGP TIEs due to normal flooding behavior and hence loss of the best prefix will lead to re-evaluation of TIEs present and re-advertisement of a new best PGP.

2.4. Re-origination

A node must re-originate policy-guided prefixes and retransmit them. The node has its database of southbound policy-guided prefixes to send in its S-TIE and its database of northbound policy-guided prefixes to send in its N-TIE.

Of course, a leaf does not need to re-originate southbound policy-guided prefixes.

2.5. Reachability Computation with PGP Consideration

During reachability computation, after prefixes are attached as specified in section 5.2.6 "Attaching Prefixes" of the RIFT base specification, PGPs are considered.

Each policy-guided prefix P has its cost and next_hop_set already stored in the associated database, as specified in Section 2.3; the cost stored for the PGP is already updated to considering the cost of the link to the advertising neighbor. By definition, a policy-guided prefix is preferred to a regular prefix.

```
    for each policy-guided prefix P:
      if P not in route_database:
         add (P, type=PolicyGuided, P.cost, next_hop_set)
         end if
      if P in route_database :
          if (route_database[P].type is not PolicyGuided) or
             (route_database[P].cost > P.cost):
            update route_database[P] with (P, PolicyGuided, P.cost,
next_hop_set)
          else if route_database[P].cost == P.cost
            update route_database[P] with (P, PolicyGuided, P.cost,
               merge(next_hop_set, route_database[P].next_hop_set))
          else
            // Not preferred route so ignore
            end if
          end if
      end for
```

Figure 1: Adding Routes from Policy-Guided Prefixes

Notice that a policy-guided prefix is always preferred to a regular prefix, even if the policy-guided prefix has a larger cost.

PGPs may overlap with prefixes introduced by automatic de-aggregation. The topic is under further discussion. The break in connectivity that leads to infeasibility of a PGP is mirrored in adjacency tear-down and according removal of such PGPs. Nevertheless, the underlying link-state flooding will be likely reacting significantly faster than a hop-by-hop redistribution and with that the preference for PGPs may cause intermittent black-holes.

3. Security Considerations

To be provided.

4. Acknowledgements

5. Normative References

[I-D.ietf-rift-rift]
Team, T., "RIFT: Routing in Fat Trees", draft-ietf-rift-rift-05 (work in progress), April 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Alia Atlas
Individual

EMail: akatlas@gmail.com

Zhaohui Zhang
Juniper Networks

EMail: zzhang@juniper.net