Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

B. Decraene
Orange

March 9, 2015

BGP Next-Hop Capabilities
draft-decraene-idr-next-hop-capability-00

Abstract

RFC 5492 defines capabilities advertisement for the BGP peer. In addition, it's useful to know the capabilities of the BGP Next-Hop, in particular for forwarding plane features. RFC 5492 is not applicable because the BGP peer may be different from the BGP Next-Hop, in particular when BGP Route Reflection is used. This document defines a mechanism to advertise such BGP Next Hop Capabilities.

This document defines a new BGP non-transitive attribute to carry Next-Hop Capabilities. This attribute is deleted when the BGP Next Hop is changed.

This document also defines a Next-Hop capability to advertise the ability to handle the Entropy Label defined in RFC 6790.

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 RFC 2119 [RFC2119].

Status of This Memo

1. Introduction

[RFC5492] defines capabilities advertisement for the BGP peer. It's also useful to know the capabilities of the BGP Next-Hop, in particular for forwarding plane features. RFC 5492 is not applicable because the BGP peer may be different from the BGP Next-Hop, in particular when BGP Route Reflection is used. This document defines a mechanism to advertise such BGP Next Hop Capabilities.

This document defines a new BGP non-transitive attribute to carry Next-Hop Capabilities. This attribute is deleted when the BGP Next Hop is changed.

This document also defines a first application to advertise the capability to handle the Entropy Label defined in [RFC6790]. Note that RFC 6790 had originally defined a BGP attribute for this but it has been latter deprecated in [RFC7447]

2. BGP Next-Hop Capability Attribute

The BGP Next-Hop Capabilities Attribute is an optional, non-transitive BGP Attribute, of value TBD1. The attribute consists of a set of Next-Hop Capabilities. Inclusion of a Next-Hop Capability "X" indicates that the BGP Next-Hop, encoded in either the NEXT_HOP attribute defined in [RFC4271] or the Network Address of Next Hop field of the MP_REACH_NLRI attribute defined in [RFC4760], supports the capability "X". This document do not make distinction between these two Next-Hop fields and refer to them as BGP Next-Hop.

A Next-Hop Capability is triple (Capability Code, Capability Length, Capability Value) aka a TLV:

A Next-Hop Capability.

```
+-----------------------------+
| Capability Code (1 octet)   |
+-----------------------------+
| Capability Length (1 octet) |
+-----------------------------+
| Capability Value (variable) |
~                             ~
+-----------------------------+
```

Capability Code : a one-octet unsigned binary integer which indicates the type of "Next-Hop Capability" advertised and unambiguously identifies an individual capability.

Capability Field: a one-octet unsigned binary integer which indicates the length, in octets, of the Value Field. A length of 0 indicates that no Value Field is present.

Value Field: a variable-length field from 0 to 255 octets. It is interpreted according to the value of the Capability Code field.

BGP speakers SHOULD NOT include more than one instance of a Next-Hop capability with the same Capability Code, Capability Length, and Capability Value. Note, however, that processing of multiple instances of such capability does not require special handling, as additional instances do not change the meaning of the announced capability; thus, a BGP speaker MUST be prepared to accept such multiple instances.

BGP speakers MAY include more than one instance of a capability (as identified by the Capability Code) with non-zero Capability Length field, but with different Capability Value and either the same or different Capability Length. Processing of these capability instances is specific to the Capability Code and MUST be described in the document introducing the new capability.

3. BGP Next-Hop Capabilities Attribute Operation

The BGP Attribute being non-transitive, as per [RFC4271], a BGP speaker which does not understand it will quietly ignore it and not pass it along to other BGP peers.

A BGP speaker which understand the BGP Next-Hop Attribute and does not change the BGP Next-Hop, SHOULD NOT change the BGP Next-Hop Attribute and SHOULD pass the attribute unchanged along to other BGP peers.

A BGP speaker which understand the BGP Next-Hop Attribute and change the BGP Next-Hop, MUST remove the received BGP Next-Hop Attribute before propagating the BGP UPDATE other BGP peers. It MAY attach a new BGP Next-Hop attribute describing the capabilities of the new BGP Next-Hop.

4. BGP Next-Hop Attribute Error Handling

A BGP Next-Hop Capability Attribute is considered malformed if the length of the Attribute is not equal to the sum of all (BGP Hop Capability Length +2) of each capability carrier in this attribute. Note that "2" is the length of the fields "Type" and "Length" of each BGP Next Hop Capability.

A BGP UPDATE message with a malformed BGP Next-Hop Capability Attribute SHALL be handled using the approach of "attribute discard" defined in [I-D.ietf-idr-error-handling]. [Note: To be Discussed. Treat as withdraw would be safer if one implementation allow changing route preference based on BGP Next-Hop Capability. But this is the case of any attribute.]

If a Next-Hop Capability is malformed, this Next-Hop Capability Type MUST be ignored. Others Next-Hop Capabilities MUST be processed as usual.[Note: To be Discussed]

5. Entropy Label Next-Hop Capability

The Entropy Label Next-Hop Capability have type code 1 and a length of 0 octet.

The inclusion of the "Entropy Label" Next-Hop Capability indicates that the BGP Next-Hop can be sent packets with a MPLS entropy label as an egress LSR for all routes in that NLRI.

A BGP speaker S that originates an UPDATE MAY include the Entropy Label Next-Hop Capability only if either of the following is true:

A1: S sets the BGP NEXT_HOP attribute to itself AND S can process entropy labels. In other words, the BGP Next-Hop can process entropy labels.

A2: S sets the BGP NEXT_HOP attribute to itself AND S swaps the advertised label without popping the advertised label(s) stack AND S knows that the egress can process the entropy label (typically when redistributing a route received with the indication that the egress can be sent the entropy label (e.g. received via BGP with the "Entropy Label" Next-Hop Capability attached, or via LDP with the ELC

TLV...). In other words, the BGP Next-Hop may or may not be able to process entropy labels, but it will not have to process it.

6. IANA Considerations

6.1. Next-Hop Capabilities Attribute

IANA is requested to allocate a new Path Attribute, called "Next-Hop Capabilities", type Code TBD1, from the "BGP Path Attributes" registry.

6.2. Next-Hop Capability registry

The IANA is requested to create and maintain a registry entitled "Next-Hop Capabilities".

The registration policies [RFC5226] for this registry are:

1-63	IETF Review
64-127	First Come First Served

IANA is requested to make the following initial assignments:

Registry Name: Next-Hop Capability.

Value	Meaning	Reference
0	Reserved	This document
1	Entropy Label	This document
2-127	Unassigned	
128-255	Private Use	This document

7. Security Considerations

This document does not introduce new security vulnerabilities in BGP. Please refer to the Security Considerations section of [RFC4271] for security mechanisms applicable to BGP.

8. Acknowledgement

The Entropy Label Next-Hop Capability defined in this document is based on the ELC BGP attribute defined in section 5.2 of [RFC6790].

9. References

9.1. Normative References

[I-D.ietf-idr-error-handling]
Chen, E., Scudder, J., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", draft-ietf-idr-error-handling-18 (work in progress), December 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

9.2. Informative References

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, February 2009.

[RFC7447] Scudder, J. and K. Kompella, "Deprecation of BGP Entropy Label Capability Attribute", RFC 7447, February 2015.

Author's Address

Bruno Decraene
Orange

Email: bruno.decraene@orange.com

Network Working Group
Internet-Draft
Updates: 6790 (if approved)
Intended status: Standards Track
Expires: July 30, 2017

B. Decraene
Orange
K. Kompella
Juniper Networks, Inc.
W. Henderickx
Nokia

January 26, 2017

BGP Next-Hop dependant capabilities
draft-decraene-idr-next-hop-capability-03

Abstract

RFC 5492 defines capabilities advertisement for the BGP peer. In addition, it is useful to be able to advertise BGP Next-Hop dependant capabilities, in particular for forwarding plane features. RFC 5492 is not applicable because the BGP peer may be different from the BGP Next-Hop, in particular when BGP Route Reflection is used. This document defines a mechanism to advertise such BGP Next Hop dependant Capabilities.

This document defines a new BGP non-transitive attribute to carry Next-Hop Capabilities. This attribute is deleted or possibly modified when the BGP Next Hop is changed.

This document also defines a Next-Hop capability to advertise the ability to handle the MPLS Entropy Label defined in RFC 6790. It updates RFC 6790 with regard to this BGP signaling.

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 RFC 2119 [RFC2119].

Status of This Memo

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 July 30, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
2. BGP Next-Hop dependant Capabilities Attribute 3
2.1. Encoding 3
2.2. Attribute Operation 4
2.3. Capability Code Operation 5
2.4. Attribute Error Handling 5
3. Entropy Label Next-Hop dependant Capability 6
3.1. Entropy Label Next-Hop Capability error handling 7
4. IANA Considerations 7
4.1. Next-Hop Capabilities Attribute 7
4.2. Next-Hop Capability registry 7
5. Security Considerations 8
6. Acknowledgement 8
7. References 8
7.1. Normative References 8
7.2. Informative References 9
Authors' Addresses 9

1. Introduction

[RFC5492] defines capabilities advertisement for the BGP peer. In addition, it is useful to be able to advertise BGP Next-Hop dependant capabilities, in particular for forwarding plane features. RFC 5492 is not applicable because the BGP peer may be different from the BGP Next-Hop, in particular when BGP Route Reflection is used. This

document defines a mechanism to advertise such BGP Next Hop Capabilities.

This document defines a new BGP non-transitive attribute to carry Next-Hop Capabilities. When the BGP Next Hop is changed, this attribute is deleted or possibly modified to take into account the capabilities of the new BGP Next-Hop. Hence it allows advertising capabilities which are dependent of the BGP Next-Hop.

This attribute advertises the capabilities of the BGP Next-Hop for the NLRI advertised in the same BGP update. A BGP Next-Hop may advertise different capabilities for different set of NLRI.

This document also defines a first application to advertise the capability to handle the MPLS Entropy Label defined in [RFC6790]. Note that RFC 6790 had originally defined a BGP attribute for this but it has been latter deprecated in [RFC7447].

2. BGP Next-Hop dependant Capabilities Attribute

2.1. Encoding

The BGP Next-Hop dependant Capabilities Attribute is an optional, non-transitive BGP Attribute, of value TBD1. The attribute consists of a set of Next-Hop Capabilities.

The inclusion of a Next-Hop Capability "X" in a BGP UPDATE message, indicates that the BGP Next-Hop, encoded in either the NEXT_HOP attribute defined in [RFC4271] or the Network Address of Next Hop field of the MP_REACH_NLRI attribute defined in [RFC4760], supports the capability "X" for the NLRI advertised in this BGP UPDATE.

This document does not make a distinction between these two Next-Hop fields and uses the term 'BGP Next-Hop' to refer to whichever one is used in a given BGP UPDATE message.

A Next-Hop Capability is a triple (Capability Code, Capability Length, Capability Value) aka a TLV:

```
+-----------------------------+
| Capability Code (1 octet)   |
+-----------------------------+
| Capability Length (1 octet) |
+-----------------------------+
| Capability Value (variable) |
~                             ~
+-----------------------------+
```

Figure 1: BGP Next-Hop Capability

Capability Code: a one-octet unsigned binary integer which indicates the type of "Next-Hop Capability" advertised and unambiguously identifies an individual capability.

Capability Length: a one-octet unsigned binary integer which indicates the length, in octets, of the Capability Value field. A length of 0 indicates that no Capability Value Field is present.

Capability Value: a variable-length field from 0 to 255 octets. It is interpreted according to the value of the Capability Code.

BGP speakers SHOULD NOT include more than one instance of a Next-Hop capability with the same Capability Code, Capability Length, and Capability Value. Note, however, that processing of multiple instances of such capability does not require special handling, as additional instances do not change the meaning of the announced capability; thus, a BGP speaker MUST be prepared to accept such multiple instances.

BGP speakers MAY include more than one instance of a capability (as identified by the Capability Code) with non-zero Capability Length field, but with different Capability Value and either the same or different Capability Length. Processing of these capability instances is specific to the Capability Code and MUST be described in the document introducing the new capability.

2.2. Attribute Operation

The BGP Next-Hop dependant Capabilities attribute being non-transitive, as per [RFC4271], a BGP speaker which does not understand it will quietly ignore it and not pass it along to other BGP peers.

A BGP speaker that understands the BGP Next-Hop dependant Capabilities Attribute and does not change the BGP Next-Hop, SHOULD NOT change the BGP Next-Hop dependant Capabilities Attribute and SHOULD pass the attribute unchanged along to other BGP peers.

A BGP speaker that understands the BGP Next-Hop dependant Capabilities Attribute and changes the BGP Next-Hop, MUST remove the received BGP Next-Hop dependant Capabilities Attribute before propagating the BGP UPDATE to other BGP peers. It MAY attach a new BGP Next-Hop dependant Capabilities attribute describing the capabilities of the new BGP Next-Hop for these NLRIs.

2.3. Capability Code Operation

A BGP speaker receiving a BGP Next-Hop Capability Code that it supports behave as defined in the document defining this Capability Code. A BGP speaker receiving a BGP Next-Hop Capability Code that it does not support MUST ignore this BGP Next-Hop Capability Code. In particular, this MUST NOT be handled as an error. In both cases, the BGP speaker MUST examine the remaining BGP Next-Hop Capability Code(s) that may be present in the BGP Next-Hop Capabilities Attribute.

The BGP Next-Hop Capability Code MUST reflect the capability of the router indicated in the BGP Next-Hop, for the NLRI advertised in the BGP UPDATE. If a BGP speaker sets the BGP Next-Hop to an address of a different router (e.g. R), it MUST NOT advertise BGP Next-Hop Capabilities not supported by this router R for these NLRI.

The presence of a Next-Hop Capability SHOULD NOT influence route selection or route preference of a route, unless tunneling is used to reach the BGP Next-Hop or the selected route has been learnt from EBGP (i.e. the Next-Hop is in a different AS). Indeed, it is in general impossible for a node to know that all BGP routers of the Autonomous System (AS) will understand a given Next-Hop Capability; and having different routers, within an AS, use a different preference for a route, may result in forwarding loops if tunnelling is not used to reach the BGP Next-Hop.

An implementations MAY allow, by configuration, removing this attribute or specific Next-Hop capabilities when advertising the routes over EBGP.

2.4. Attribute Error Handling

A BGP Next-Hop dependant Capabilities Attribute is considered malformed if the length of the Attribute is not equal to the sum of all (BGP Next-Hop dependant Capability Length +2) of the capabilities carried in this attribute. Note that "2" is the length of the fields "Type" and "Length" of each BGP Next Hop dependant Capability, as the capability length only account for the length of the Value field.

A document that specifies a new Next-Hop Capability SHOULD provide specifics regarding what constitutes an error for that Next-Hop Capability.

A BGP UPDATE message with a malformed BGP Next-Hop dependant Capabilities Attribute SHALL be handled using the approach of "attribute discard" defined in [RFC7606].

Unknown Next-Hop Capabilities Codes MUST NOT be considered as an error. They MUST be silently ignored.

If a Next-Hop dependant Capability is malformed, this Next-Hop Capability Type MUST be ignored. Others Next-Hop Capabilities MUST be processed as usual.

3. Entropy Label Next-Hop dependant Capability

The Entropy Label Next-Hop Capability has type code 1 and a length of 0 or 1 octet.

The inclusion of the "Entropy Label" Next-Hop Capability indicates that the BGP Next-Hop can be sent packets, for all routes indicated in the NRLI, with a MPLS entropy label (ELI, EL) added immediately after the label stack advertised with the NLRI.

On the receiving side, suppose BGP speaker S has determined that packet P is to be forwarded according to BGP route R, where R is a route of one of the labeled address families. And suppose that L is the label stack embedded in the NLRI of route R. Then to forward packet P according to route R, S either replaces P’s top label with L, or else pushes L onto the MPLS label stack. If the EL-Capability is advertised in the BGP UPDATE advertising this route R, S knows that it may safely place the ELI and an EL on the label stack immediately beneath L.

A BGP speaker S that sends an UPDATE with the BGP Next-Hop "NH" MAY include the Entropy Label Next-Hop Capability only if the NLRI are labelled and for all the NLRI in the BGP UPDATE, either of the following is true:

- o Egress case: NH is the egress of the LSP advertised with the NLRI and its capable of handling the ELI during the lookup of the MPLS top label.

- o Transit LSR case: NH is a transit LSR for the LSP advertised with the NLRI (i.e. NH swaps one of the label advertised in the NLRI) and next downstream BGP Next-Hop(s) has(have) advertised the Entropy Label Next-Hop Capability (or a similar capability

signalled by protocol P if the route is redistributed, by NH, from protocol P to BGP).

3.1. Entropy Label Next-Hop Capability error handling

If the Entropy Label Next-Hop Capability is present more than once, it MUST be considered as received once with a length of 0.

If the Entropy Label Next-Hop Capability is received with a length other than 0 or 1, it is not considered malformed, but its semantics are exactly the same as if it had a length of 1. In other words, additional octets MUST be ignored. This is to allow for graceful future extension.

4. IANA Considerations

4.1. Next-Hop Capabilities Attribute

IANA is requested to allocate a new Path Attribute, called "Next-Hop Capabilities", type Code TBD1, from the "BGP Path Attributes" registry.

4.2. Next-Hop Capability registry

The IANA is requested to create and maintain a registry entitled "Next-Hop Capabilities".

The registration policies [RFC5226] for this registry are:

1-63	IETF Review
64-127	First Come First Served
128-250	Standards Action
251-254	Experimental Use
255	Reserved

IANA is requested to make the following initial assignments:

Registry Name: Next-Hop Capability.

Value	Meaning	Reference
0	Reserved (not to be allocated)	This document
1	Entropy Label	This document
2-250	Unassigned	
251-254	Experimental	This document
255	Reserved (for futur registry extension)	This document

5. Security Considerations

This document does not introduce new security vulnerabilities in BGP. Specifically, an operator who is relying on the information carried in BGP must have a transitive trust relationship back to the source of the information. Specifying the mechanism(s) to provide such a relationship is beyond the scope of this document. Please refer to the Security Considerations section of [RFC4271] for security mechanisms applicable to BGP.

6. Acknowledgement

The Entropy Label Next-Hop Capability defined in this document is based on the ELC BGP attribute defined in section 5.2 of [RFC6790].

The authors wish to thank John Scudder for the discussions on this topic and Eric Rosen for his in-depth review of this document.

The authors wish to thank Jie Dond for his review and comments.

7. References

7.1. Normative References

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

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <http://www.rfc-editor.org/info/rfc4271>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <http://www.rfc-editor.org/info/rfc4760>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <http://www.rfc-editor.org/info/rfc6790>.

[RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <http://www.rfc-editor.org/info/rfc7606>.

7.2. Informative References

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, DOI 10.17487/RFC5492, February 2009, <http://www.rfc-editor.org/info/rfc5492>.

[RFC7447] Scudder, J. and K. Kompella, "Deprecation of BGP Entropy Label Capability Attribute", RFC 7447, DOI 10.17487/RFC7447, February 2015, <http://www.rfc-editor.org/info/rfc7447>.

Authors' Addresses

Bruno Decraene
Orange

Email: bruno.decraene@orange.com

Kireeti Kompella
Juniper Networks, Inc.
1194 N. Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: kireeti.kompella@gmail.com

Wim Henderickx
Nokia
Copernicuslaan 50
Antwerp 2018, CA 95134
Belgium

Email: wim.henderickx@nokia.com

IDR
Internet Draft
Intended status: Standards Track
Expires: September 10, 2015

Luyuan Fang
Microsoft
Chandra Ramachandran
Juniper Networks
Fabio Chiussi
Cisco Systems
Yakov Rekhter

March 9, 2015

BGP-LU for HSDN Label Distribution
draft-fang-idr-bgplu-for-hsdn-00

Abstract

This document describes the use of BGP Labeled Unicast (BGP-LU) with modified BGP Route Reflector (RR) operation for label distribution in the Hierarchical SDN (HSDN) control plane for the hyper-scale Data Center (DC) and cloud networks.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Table of Contents

1. Introduction 3
2. Terminology 5
3. Label Mapping Server/RR 5
  3.1. Peer Community 6
4. BGP-LU Procedures 6
  4.1. Advertisements from UP Destination 7
  4.2. Advertisements from UPBN 8
  4.3. Advertisements from LM-RR 8
  4.4. Route Resolution 9
5. Illustration of BGP-LU Procedures 9
  5.1. UP2-1 Destinations 9
  5.2. UP1-1 Destinations 10
  5.3. UP0 Destinations 11
  5.4. Packets from Server3 to Server1 12
6. Context Labels 13
  6.1. UP2-1 Destinations 14
  6.2. UP1-1 Destinations 14
  6.3. UP0 Destinations 14
7. Security Considerations 15
8. IANA Considerations 15
9. Normative References 15
10. Informative References 15
Authors' Addresses 15

1. Introduction

Hierarchical SDN (HSDN) [I-D.fang-mpls-hsdn-for-hsdc] is an architectural solution to scale a hyper-scale cloud consisting of many Data Centers (DCs) interconnected by a Data Center Interconnect (DCI) to tens of millions of physical underlay endpoints, while efficiently handling both Equal Cost Multi Path (ECMP) load-balanced traffic and any-to-any end-to-end Traffic Engineered (TE) traffic. The HSDN reference model, operation, and requirements are described in [I-D.fang-mpls-hsdn-for-hsdc].

HSDN is designed to allow the physical decoupling of control and forwarding, and have the LFIBs configured by a controller according to a full SDN approach. Such a controller-centric approach is described in [I-D.fang-mpls-hsdn-for-hsdc].

However, HSDN is also meant to support the traditional distributed routing and label distribution protocol approach to distribute the labels. This hybrid approach may be particularly useful during technology migration. This document specifies the use of BGP Labeled Unicast (BGP-LU) for label distribution and LFIB configuration in the HSDN control plane.

In the HSDN architecture, the DC/DCI network is partitioned into hierarchical underlay partitions (UPs) such that the number of destinations in each UP does not increase beyond the limit imposed by capabilities of network nodes. Once the DC cloud has been partitioned to the desired configuration, the traffic from a source endpoint to a destination endpoint uses a stack of labels, one label per each level in the hierarchy, whose semantics indicate to the forwarding network nodes at each level which destination in its local UP should forward the packet to. The label semantics can also identify a specific path (or group of paths) in the UP, rather than simply a destination.

In other words, the label stack indirectly represents the UPs that the packet should traverse to reach the destination end device.

```
                                 UP0
     \ +---------+   +---------+           +---------+   +---------+ /
      \|UPBN1-1-1|~~~|UPBN1-1-2|-----------|UPBN1-2-1|~~~|UPBN1-2-2|/
       +---------+   +---------+           +---------+   +---------+
      (                         )         (                         )
     (           UP1-1           )       (           UP1-2           )
      (                         )         (                         )
       +---------+   +---------+           +---------+   +---------+
       |UPBN2-1-1|~~~|UPBN2-1-2|           |UPBN2-2-1|~~~|UPBN2-2-2|
       +---------+   +---------+           +---------+   +---------+
      (                         )         (                         )
     (          UP2-1            )       (           UP2-2           )
      (                         )         (                         )
       +---------+   +---------+           +---------+   +---------+
       | Server1 |~~~| Server2 |           | Server3 |~~~| Server4 |
       +---------+   +---------+           +---------+   +---------+
```

Figure 1 - Example topology with 2 levels of partitioning

Considering the example partitioning in Figure 1, which has 3 levels in the hierarchy, packets from Device3 to Device1 require 3 Path Labels (PLs).

- Top label (PL0) will forward the packet to one of the UPBN1-1 nodes, which are grouped as UPBG1-1 (which is a destination in UP0)

- Next label (PL1) will forward the packet to one of the UPBN2-1 nodes, which are grouped as UPBG2-1 (which is a destination in UP1-1)

- Next label (PL2) will forward the packet to Device1 (which is a destination in UP2-1)

This document proposes BGP-LU based procedures for:

- How UPBN learns the destinations in its UP and the label that should be installed in LFIB to forward traffic to these destinations

- How UPBN learns the context labels used by other UPBN destinations in the partition if the DC operator implements a policy of using locally assigned labels on UPBNs

The procedures specified in the document are applicable to ECMP traffic in HSDN based DC cloud architectures.

2. Terminology

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 RFC 2119 [RFC2119].

This document inherits the terminology defined in [I-D.fang-mpls-hsdn-for-hsdc] and additionally introduces the following terms that apply when BGP-LU based control plane is used to realize HSDN architecture.

- o RR: BGP Route Reflector.

- o BGP Peer Group: Collection of BGP peers for which a set of policies are applied on a BGP speaker.

- o Label Mapping Server: A node present in each Underlay Partition that allocates labels for destinations in the partition.

- o Label Mapping RR (LM-RR): A modified or customized BGP RR that uses BGP-LU to advertise label bindings for destinations in UP. In other words, Label Mapping RR is an implementation of Label Mapping Server that uses BGP-LU to advertise the labels for partition destinations.

- o Peer Community: An IP based extended community carried in BGP update that represents UPBG of a partition.

- o Route Resolver: A single or a collection of entities that provides the MPLS label stack to reach a destination underlay end device.

3. Label Mapping Server/RR

This document specifies modifications to BGP Route Reflection procedures defined in [RFC4456] in order to provide a mechanism for Label Mapping Server of a UP to distribute labels for UP destinations. These modified procedures are applicable to both BGP RR server as well as the RR client and implementations must activate these procedures based on user-configured policy.

- When LM-RR receives BGP-LU advertisement [RFC3107] (i) whose NLRI and BGP next-hop are the same, and (ii) that contains a valid Peer-Community (specified in Section 3.1), then LM-RR looks up its local database of known Peer community values. If the Peer-Community value is new, then LM-RR adds the newly learnt Peer-community in the database.

- When LM-RR receives IP route (IP version 4 AFI) whose (i) NLRI and BGP next-hop are different, and (ii) BGP next-hop belongs to a known Peer-Community from its local database, then LM-RR performs the following actions. Otherwise, LM-RR behaves like vanilla BGP RR specified in [RFC4456].

  - o LM-RR checks whether a label has been allocated for {UP destination, Peer-Community} pair. If not, LM-RR allocates a label for {UP destination, Peer-Community} and let us call this label PL1.

  - o LM-RR originates a BGP-LU advertisement for the same IP destination containing PL1 in the L-BGP NLRI but does not modify the BGP next-hop attribute from the received IP route. In other words, the modified LM-RR procedures result in LM-RR effectively reflecting a BGP-LU route in response to a vanilla IP route when the above specified conditions are met.

Conceptually, the above procedures result in LM-RR (implementing Label Mapping Server of a UP) allocating a "partition-unique" label for every destination in the partition, and all UPBNs of the partition forwarding MPLS packet with that label to the particular destination in the partition. It should be noted that LM-RR of a UP may allocate labels having any structure that reflects the UP hierarchy as specified in Section 5 of [I-D.fang-mpls-hsdn-for-hsdc].

In other words, while the label allocated by LM-RR is strictly "partition- unique", the DC operator may apply a label allocation policy that would result in same label allocated for same destination across partitions (which may be true for destinations that are present "up" the hierarchy).

3.1. Peer Community

This document introduces a new extended community that enables the receiving iBGP speaker to group the iBGP peers into a community. The application of new extended community in this document is that it allows the receiving BGP speaker to determine which iBGP peers belong to a UPBG.

The details of the new extended community are TBD.

4. BGP-LU Procedures

The BGP-LU based control plane mechanism specified in this document assumes the following set of policies be applied on various network nodes. The policy configurations required are as follows.

- Each UPBN of a UP is configured to be a BGP RR and all UP destinations are configured as clients of UPBN. That is, each end-device in the lowest level UP has one iBGP peering session with each UPBN of the UP. Note that it is not essential that UPBNs are configured as BGP RRs and the same outcomes described from the procedures below may be arrived by not configuring UPBNs as RRs.

- Each UPBN of a non-UP0 partition is also connected to its higher level partition. For example, UPBNj of UPj will be connected to UPi where i=j-1. UPBNj has one iBGP peering session with each UPBNi, and one iBGP peering session with UPj destinations. Note that on UPBNj, iBGP peering sessions with UPBNi and UPj destinations are configured to be in different BGP peer groups.

- UPBNj has a policy to automatically export destinations learnt from UPBNi peer group to UPj peer group (where i=j-1). But UPBNj does not export destinations learnt from UPj peer group to UPBNi peer group. This export policy on UPBNj limits the number of BGP advertisements that any network node in UPi has to process apart from limiting the number of LFIB entries in network nodes.

- If UPBNs learn destinations in its UP and its parent UP using IGP, then UPBNj runs two separate IGP routing instances one corresponding to UPj and one corresponding to UPi (where i=j-1). UPBNj does not leak any route between the IGP instances. Alternatively, UPBNj may learn destinations in UPj and UPi using other mechanisms and such mechanisms are outside the scope of this document.

- Each Underlay Partition (UP) has one Label Mapping RR (or LM-RR) that is responsible for advertising the labels allocated for the destinations in that UP. Each UPBN of the UP is configured with UPBG that it belongs to and has iBGP peering session with LM-RR.

- LM-RR also maintains iBGP peering session(s) with Route Resolver. How Router Resolver is realized is outside the scope of this document, but for the purpose of this document it should be noted that LM-RR of a UP uses this iBGP peering session(s) to advertise the labels allocated for UP destinations to Resolver.

4.1. Advertisements from UP Destination

- Each End-device in a UP advertises a "vanilla" IP route in BGP to all UPBNs of the UP (UPBNs are configured as RRs while UP destinations are configured as clients). The advertisement contains NLRI and BGP nexthop attribute set to the address of the end-device that advertises the route.

- Each UPBNj that is a destination in UPi (where i=j-1) advertises a BGP-LU route with NULL label to UPBNi. The advertisement contains NLRI and BGP nexthop attribute set to the address of UPBNj.

  - o For UPi, the destinations may be UPBNj where j=i-1.

  - o BGP-LU routes originated by UPBNj will have "Peer- Community" appended to the route where the "Peer-Community" corresponds to UPBGj that UPBNj belongs to.

4.2. Advertisements from UPBN

- When UPBN receives the advertisements from UP destinations (as specified in Section 4.1), UPBN re-advertises these UP destinations to LM-RR.

  - o Each UPBN in the UP also originates a "vanilla" IP route corresponding to each destination in the UP

  - o UPBN removes the "Peer-Community" corresponding to UPBNj if present in the advertisement received from UP destination.

  - o If the UP destination is learnt from a BGP-LU route, the UPBN removes the label in its re-advertisement to LM-RR.

- UPBN also originates a BGP-LU route with NULL label for itself i.e. with NLRI and BGP nexthop attribute set to self

  - o UPBNi routes will have "Peer-Community" appended where the "Peer-Community" corresponds to UPBGi that UPBN belongs to.

4.3. Advertisements from LM-RR

- When LM-RR receives a BGP advertisement that contains "Peer-Community" (that denotes a UPBG), it allocates a label from its label space if a label is not already allocated for the destinations advertised by peers that belong to the UPBG.

  - o LM-RR determines whether the originator is a UPBN by using the route already advertised by the UPBN with "Peer- Community".

  - o LM-RR allocates one label per UPBG per destination. As all UPBNs of a UP belong to one UPBG, LM-RR will allocate N labels for a UP with N destinations.

- LM-RR originates in the UP a BGP-LU route for each "vanilla" IP route learnt from UPBN (see Section 4.2).

- o Label value that is set in BGP-LU route is equal to the label LM-RR has allocated for the UP destination per UPBG.

- o LM-RR retains the BGP nexthop attribute present in the "vanilla" IP advertisement.

- o LM-RR also advertises the BGP-LU route to Route Resolver in order to provide enough information to Route Resolver to recursively resolve a remote End-device destination. Note that the method of realizing Route Resolver is beyond the scope of this document.

- When UPBN receives the BGP-LU route (per UP destination) originated from LM-RR, it installs the label in its LFIB and sets up the LFIB entry to forward MPLS packet received with that label to the specific UP destination.

  - o If the UP destination has been learnt via a "vanilla" IP route without an associated "Peer-Community", then all ECMP paths in the LFIB entry terminate at the UP destination (which is an End-device in this case).

  - o If the UP destination has been learnt via a BGP-LU route with a corresponding "Peer-Community" then the ECMP paths in the LFIB entry terminate at all lower level UPBNs having the same "Peer-Community".

4.4. Route Resolution

As a consequence of the procedures described in Section 4.1 to 4.3, Route Resolver of the DC cloud will have the knowledge of the destinations in all UPs and the UPBNs that have advertised those UP destinations. Route Resolver uses this information to construct MPLS label stack to forward the packet to desired destination End-device.

The mechanism with which Route Resolver is implemented in the DC cloud is outside the scope of this document.

5. Illustration of BGP-LU Procedures

This sections provides an example of the procedures described in Section 4 using the example topology provided in Figure 1.

5.1. UP2-1 Destinations

UPBN2-1-1 originated routes:

{Server1, NH: UPBN2-1-1}

{Server2, NH: UPBN2-1-1}

{UPBN2-1-1, NH: UPBN2-1-1, Label: NULL, Peer-Community: UPBG2-1}

UPBN2-1-2 originated routes:

{Server1, NH: UPBN2-1-2}

{Server2, NH: UPBN2-1-2}

{UPBN2-1-2, NH: UPBN2-1-2, Label: NULL, Peer-Community: UPBG2-1}

LM-RR-2-1 originated routes:

{Server1, NH: UPBN2-1-1, Label: PL2-1}

{Server1, NH: UPBN2-1-2, Label: PL2-1}

{Server2, NH: UPBN2-1-1, Label: PL2-2}

{Server2, NH: UPBN2-1-2, Label: PL2-2}

Note that LM-RR has to advertise these routes to Route Resolver also. The method of realizing Route Resolver is beyond the scope of this document.

5.2. UP1-1 Destinations

UPBN1-1-1 originated routes:

{UPBN2-1-1, NH: UPBN1-1-1}

{UPBN2-1-2, NH: UPBN1-1-1}

{UPBN1-1-1, NH: UPBN1-1-1, Label: NULL, Peer-Community: UPBG1-1}

UPBN1-1-2 originated routes:

{UPBN2-1-1, NH: UPBN1-1-2}

{UPBN2-1-2, NH: UPBN1-1-2}

{UPBN1-1-2, NH: UPBN1-1-2, Label: NULL, Peer-Community: UPBG1-1}

LM-RR-1-1 originated routes:

{UPBN2-1-1, NH: UPBN1-1-1, Label: PL1-1}

{UPBN2-1-1, NH: UPBN1-1-2, Label: PL1-1}

{UPBN2-1-2, NH: UPBN1-1-1, Label: PL1-1}

{UPBN2-1-2, NH: UPBN1-1-2, Label: PL1-1}

Note that same label PL1-1 is assigned because UPBN2-1-1 and UPBN2-1-2 belong to the same UPBG2-1.

5.3. UP0 Destinations

UPBN1-1-1 originated routes:

{UPBN1-1-1, NH: UPBN1-1-1, Label: NULL, Peer-Community: UPBG1-1}

UPBN1-1-2 originated routes:

{UPBN1-1-1, NH: UPBN1-1-1, Label: NULL, Peer-Community: UPBG1-1}

UPBN1-2-1 originated routes:

{UPBN1-1-1, NH: UPBN1-2-1}

{UPBN1-1-2, NH: UPBN1-2-1}

{UPBN1-2-1, NH: UPBN1-2-1, Label: NULL, Peer-Community: UPBG1-2}

UPBN1-2-2 originated routes:

{UPBN1-1-1, NH: UPBN1-2-2}

{UPBN1-1-2, NH: UPBN1-2-2}

{UPBN1-2-2, NH: UPBN1-2-2, Label: NULL, Peer-Community: UPBG1-2}

LM-RR-0 originated routes:

{UPBN1-1-1, NH: UPBN1-2-1, Label: PL0-1}

{UPBN1-1-2, NH: UPBN1-2-1, Label: PL0-1}

{UPBN1-1-1, NH: UPBN1-2-2, Label: PL0-1}

{UPBN1-1-2, NH: UPBN1-2-2, Label: PL0-1}

Note that same label PL0-1 is allocated because UPBN1-1-1 and UPBN1-1-2 belong to the same UPBG1-1.

5.4. Packets from Server3 to Server1

1. When Server3 wants to send traffic to Server1, it requests the "Route Resolver" for the label stack and immediate nexthop to send the packet to.

2. Resolver will return the label stack {PL0-1, PL1-1, PL2-1} and nexthop of UPBN1-2-1. As the UPBN destinations in higher level are advertised into lower levels, Server3 can be assumed to have enough information to forward the packet to UPBN1-2-1.

   a. By configuring UPBN1-2-1 to advertise its address as FEC in UP1-2 that is in turn leaked into UP2-2, Server3 can be assumed to have a LSP to UPBN1-2-1 to forward the MPLS packet having label stack {PL0-1, PL1-1, PL2-1}

   b. Note that Resolver may also return two nexthops UPBN1-2-1 and UPBN1-2-2 along with the label stack {PL0-1, PL1-1, PL2- 1}.

   c. Alternatively, UPBN2-2-1 and UPBN2-2-2 may be made to learn PL0-1 label that should be forwarded to UPBG1-2. In such a case, the Resolver need not provide nexthop information but only provides label stack {PL0-1, PL1-1, PL2-1} to Server3 and Server3 will always load balance MPLS packets to UPBNs of UP2-2.

3. From the BGP-LU route advertised learnt from UP0 (Section 5.3), UPBN1-2-1 should have already installed in LFIB an entry from PL0-1 with ECMP paths to UPBN1-1-1 and UPBN1-1-2.

   a. UPBN1-2-1 pops PL0-1 and load balances the packets to UPBN1-1-1 and UPBN1-1-2.

   b. Before forwarding the MPLS packet with label stack {PL1-1, PL2-1}, UPBN1-2-1 pushes the LSP label for UPBN1-1-1 or

UPBN1-1-2 depending on where the packet is forwarded to.

4. When UPBN1-1-1 or UPBN1-1-2 receives the packet with label stack {PL1-1, PL2-1}, either of them should have already installed an LFIB entry for PL1-1 (Section 5.2). In this example, let us assume that it is UPBN1-1-1 that has received the packet.

   a. UPBN1-1-1 receives PL1-1 as top of label stack because PHP is assumed to be in place.

   b. UPBN1-1-1 pops PL1-1 and load balances the packets UPBN2-1- 1 and UPBN2-1-2.

   c. Before forwarding the MPLS packet with label stack {PL2-1}, UPBN1-1-1 pushes the LSP label for UPBN2-1-1 or UPBN2-1-2 depending on where the packet is forwarded to.

5. When UPBN2-1-1 or UPBN2-1-2 receives the packet with label stack {PL2-1}, either of them should have already installed an LFIB entry for PL2-1 (Section 5.1). In this example, let us assume it is UPBN2-1-1 that has received the packet.

   a. UPBN2-1-1 pops PL2-1 and forwards the packet to Server1.

   b. Before forwarding the packet, UPBN2-1-1 pushes the LSP label for Server1.

6. Server1 receives the packet that only contains VL if any corresponding to the overlay service instance.

6. Context Labels

The label allocated by LM-RR for a UP destination is "partition-unique" and should be installed in LFIB of all UPBNs of that UP. This assumes that UPBNs of the UP are capable of setting aside some labels to avoid "partition-unique" labels from colliding with UPBN platform label space. With the following modifications to the procedures described in Section 4, this assumption is no longer required.

- If the destination in a UPi is a UPBN for a lower level UP (say UPj where j=i+1), then instead of originating BGP-LU route with NULL label in UPi it originates BGP-LU route with a non-NULL label allocated from its platform label space. The label represents the "context" label space corresponding to UPj and the LFIB entries in the context LFIB correspond to the UPj destinations (change to procedure in Section 4.1).

- LM-RR of UPi should act as vanilla BGP-RR for BGP-LU routes also

and not execute any special procedures (by allocating label) when it receives BGP-LU route from UPBNj.

The consequence of the modified procedures involving context labels is that whenever UPBN forwards MPLS packet to next UPBG, it has to push an additional label to enable the receiving UPBN (in next UPBG) to lookup in appropriate context LFIB for forwarding the packet.

The following example illustrates the changes to the advertisements to the example in Section 5 for the use of context labels.

6.1. UP2-1 Destinations

There is no change in advertisements listed in Section 5.1. This is because the UP2-1 destinations are Servers that are not UPBNs belong to any UPBG.

6.2. UP1-1 Destinations

Apart from the advertisements listed in Section 5.2, the UPBNs of UP2-1 that are the destinations in UP1-1 advertise a BGP-LU route with non-NULL label. So when UPBN1-1-1 and UPBN1-1-2 forward traffic to UPBG2-1, they push either CL2-1 or CL2-2 depending on whether the traffic is forwarded to UPBN2-1-1 or UPBN2-1-2.

UPBN2-1-1 originated routes:

{UPBN2-1-1, NH: UPBN2-1-1, Label: CL2-1, Peer-Community: UPBG2-1}

UPBN2-1-2 originated routes:

{UPBN2-1-2, NH: UPBN2-1-2, Label: CL2-2, Peer-Community: UPBG2-1}

6.3. UP0 Destinations

All BGP-LU routes that UPBNs in UP0 originate advertising their own address as destination should contain non-NUL label. The UPBNs should install LFIB entry corresponding to the label advertised to lookup in the context LFIB corresponding to LM-RR-0.

UPBN1-1-1 originated routes:

{UPBN1-1-1, NH: UPBN1-1-1, Label: CL0-1-1, Peer-Community: UPBG1-1}

UPBN1-1-2 originated routes:

{UPBN1-1-1, NH: UPBN1-1-1, Label: CL0-1-2, Peer-Community: UPBG1-1}

UPBN1-2-1 originated routes:

{UPBN1-2-1, NH: UPBN1-2-1, Label: CL0-2-1, Peer-Community: UPBG1-2}

UPBN1-2-2 originated routes:

{UPBN1-2-2, NH: UPBN1-2-2, Label: CL0-2-2, Peer-Community: UPBG1-2}

7. Security Considerations

TBD.

8. IANA Considerations

TBD

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, May 2001.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, April 2006.

10. Informative References

[I-D.fang-mpls-hsdn-for-hsdc] L. Fang, et. al., "MPLS-Based Hierarchical SDN for Hyper-Scale DC/Cloud", draft-fang-mpls-hsdn-for-hsdc-01 (work in progress), March 2015.

Authors' Addresses

Luyuan Fang
Microsoft
5600 148th Ave NE
Redmond, WA 98052
Email: lufang@microsoft.com

Chandra Ramachandran
Juniper Networks
Electra, Exora Business Park Marathahalli - Sarjapur Outer Ring Road
Bangalore, KA 560103

India
Email: csekar@juniper.net

Fabio Chiussi
Cisco
500 108th Avenue N.E., Suite 500
Bellevue, Washington 98004
Email: fchiussi@cisco.com

Yakov Rekhter

IDR
Internet Draft
Intended status: Standards Track
Expires: July 26, 2016

Luyuan Fang
Deepak Bansal
Microsoft
Chandra Ramachandran
Juniper Networks
Fabio Chiussi

Nabil Bitar
Verizon
Yakov Rekhter

January 23, 2016

BGP-LU for HSDN Label Distribution
draft-fang-idr-bgplu-for-hsdn-03

Abstract

This document describes modifications of BGP Labeled Unicast (BGP-LU) procedures for label distribution in a partitioned network. Specifically, these procedures are suitable for building the Hierarchical SDN (HSDN) control plane for the hyper-scale Data Center (DC) and cloud networks.

Table of Contents

1. Introduction 3
2. Terminology 5
3. Description of BGP-LU Procedures 7
3.1. Partitioned-Unique Label Info Extended Community 10
3.2 Partition-Unique Label Info Extended Community Procedures . 11
3.3 BGP Policies on UPBNs and LMS 13
3.4 BGP-LU Procedures for UP0 Destinations 14
3.5 Advertising labels without partition label extended community 15
4. Route Resolution in HSDN Architecture 16
5. Security Considerations 17
6. IANA Considerations 17
7. Acknowledgments 17
8. Normative References 17
9. Informative References 18
Authors' Addresses 18

1. Introduction

This document describes modifications to BGP Labeled Unicast (BGP-LU)-based procedures for label distribution [RFC3107] in a partitioned network where a label stack is used for forwarding. Current BGP-LU procedures do not provide mechanisms for distributing and installing operator-assigned partition-scope labels.

Specifically, the modifications described in this document are suitable for label distribution in the control plane of a MPLS-based Hierarchical SDN (HSDN) Data Center (DC) and cloud network.

Hierarchical SDN (HSDN) [I-D.fang-mpls-hsdn-for-hsdc] is an architectural solution to scale a hyper-scale cloud consisting of DCs interconnected by a Data Center Interconnect (DCI) to tens of millions of physical underlay endpoints, while efficiently handling both Equal Cost Multi Path (ECMP) load-balanced traffic and any-to-any end-to-end Traffic Engineered (TE) traffic. The HSDN reference model, operation, and requirements are described in [I-D.fang-mpls-hsdn-for-hsdc].

HSDN is designed to allow the physical decoupling of control and forwarding, and have the LFIBs configured by a controller according to a full SDN approach. Such a controller-centric approach is described in [I-D.fang-mpls-hsdn-for-hsdc].

However, the HSDN control plane can also be built in a hybrid approach, using a routing or label distribution protocol to distribute the labels, together with a controller. This hybrid approach may be particularly useful during technology migration. This document specifies the use of BGP-LU for label distribution and LFIB configuration in the HSDN control plane.

In the HSDN architecture, the DC/DCI network is partitioned into hierarchical underlay partitions (UPs) such that the number of destinations in each UP does not increase beyond the limit imposed by capabilities of network nodes. Once the DC cloud has been partitioned to the desired configuration, the traffic from a source endpoint to a destination endpoint uses a stack of labels, one label per each level in the hierarchy, whose semantics indicate to the forwarding network nodes at each level which destination in its local UP should forward the packet to. The label semantics can also identify a specific path (or group of paths) in the UP, rather than simply a destination.

In other words, the label stack indirectly represents the UPs that the packet should traverse to reach the destination end device. More precisely, the outer label specifies the destination in the partition at the highest level that the packet should traverse, while the other

labels specify the destination in each partition that the packet traverse thereafter.

```
                                 UP0
  \ +---------+   +---------+           +---------+   +---------+ /
   \|UPBN1-1-1|~~~|UPBN1-1-2|-----------|UPBN1-2-1|~~~|UPBN1-2-2|/
    +---------+   +---------+           +---------+   +---------+
   (                         )         (                         )
  (           UP1-1           )       (           UP1-2           )
   (                         )         (                         )
    +---------+   +---------+           +---------+   +---------+
    |UPBN2-1-1|~~~|UPBN2-1-2|           |UPBN2-2-1|~~~|UPBN2-2-2|
    +---------+   +---------+           +---------+   +---------+
   (                         )         (                         )
  (          UP2-1            )       (           UP2-2           )
   (                         )         (                         )
    +---------+   +---------+           +---------+   +---------+
    | Server1 |~~~| Server2 |           | Server3 |~~~| Server4 |
    +---------+   +---------+           +---------+   +---------+
```

Figure 1 - Example topology with 3 levels of partitioning

In the example of Figure 1, there are 3 levels in the hierarchical partitioning. The UPs are connected by a number of Underlay Partition Border Nodes (UPBNs), grouped in Underlay Partition Border Groups (UPBGs). The UPBGs are the destinations for ECMP-forwarded traffic in each partition.

Packets from Server3 to Server1 use a label stack consisting of 3 Path Labels (PLs) for forwarding.

- Top label (PL0) forwards the packet to one of the UPBN1-1 nodes, which are grouped as UPBG1-1, connecting to UP1-1, which contains Server1 (note that, by definition of HSDN forwarding, PL0 points to UPBG1-1, i.e., the destination in UP0, rather than UPBG2-1).

- Next label (PL1) forwards the packet to one of the UPBN2-1 nodes, which are grouped as UPBG2-1, connecting to UP2-1, which contains Server1 (UPNBG2-1 is a destination in UP1-1).

- Next label (PL2) forwards the packet to Server1 (which is a destination in UP2-1)

This document proposes modified BGP-LU based procedures for:

- How each UPBN learns the destinations in its UP and the operator

assigned partition unique labels that should be installed in its LFIB to forward traffic to these destinations;

- How UPBN learns the context labels used by other UPBN destinations in the partition if the DC operator implements a policy of using separate LFIBs for installing partition unique labels on UPBNs

We also introduce an associated new extended community [RFC4360] that serves the following purposes:

1. Enables a UPBN to trigger the modified BGP-LU behavior to allow distribution of partition-unique labels to UPBNs from Label Mapping Server (LMS), and

2. Identifies which LFIB partition unique labels should be installed into (if there is ambiguity due to overlapping label name spaces), and

Such extended community allows to advertise persistent labels, which can survive across BGP session restarts.

Strictly speaking, the labels advertised with the new mechanisms described in this document are not typical downstream-advertised labels, but they are more similar to upstream-advertised labels installed in context LFIBs corresponding to upstream.

It should be noted that the BGP-LU procedures specified in this document may be implemented through operator configured policy using any existing BGP community types if some conditions are met. The minor changes to the procedures and the conditions under which policy based application of an existing BGP community can be used are described in Section 3.5.

The procedures specified in the document are applicable to ECMP traffic in mpls-based HSDN DC cloud architectures.

2. Terminology

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 RFC 2119 [RFC2119].

This document inherits the terminology defined in [I-D.fang-mpls-hsdn-for-hsdc] and additionally introduces the following terms that apply when BGP-LU based control plane is used to realize HSDN architecture.

o Border Node (BN): A border node is a node that is present in a UP.

In HSDN architecture, UPBNi is a special BN that connects UPi with UPi-1.

- o Partition Label Space: Label space that is shared by all border nodes of a UP to reach a destination in the UP. For a border node, UP destinations comprise other border nodes and end devices that are present in the UP.

- o Partition Labels: Operator assigned labels that belong to partition label space corresponding to a UP. The labels need not be allocated from the platform label space on the BNs but may be directly installed in the context table corresponding the UP.

- o Label Mapping Server (LMS): A BGP speaker present in each UP that allocates labels for destinations in the partition and distributes the labels to border nodes through BGP-LU.

- o BGP Peer Group: Collection of BGP peers for which a set of policies are applied on a BGP speaker.

- o Partition-Unique Label Info Community: A new type of BGP extended community that contains the operator assigned partition unique label for the BGP destination, origin partition and border group identifier.

- o Border-group Community: This community identifies a group of border nodes that interconnect two partitions and is configured as policy on the border nodes as well as the LMS. It acts as the UPBG identifier.

- o Route Resolver: A single or a collection of entities that provides the MPLS label stack to reach a destination underlay end device.

Term	Definition
BGP	Border Gateway Protocol
BGP-LU	Border Gateway Protocol Labeled Unicast
BN	Border Node
DC	Data Center
DCI	Data Center Interconnect
ECMP	Equal Cost MultiPathing
FIB	Forwarding Information Base
HSDN	Hierarchical SDN
LFIB	Label Forwarding Information Base
LMS	Label Mapping Server
MPLS	Multi-Protocol Label Switching
SDN	Software Defined Network
UP	Underlay Partition

UPBG	Underlay Partition Border Group
UPBN	Underlay Partition Border Node
TE	Traffic Engineering

3. Description of BGP-LU Procedures

This section provides an overview of how operator assigned partition label space is used to achieve end-to-end forwarding of label stacked packets. Consider the DC network that is present in the right hand side DC in Figure 1. The diagram in Figure 2 is a part of the DCI network in Figure 1 (the partitions are arranged horizontally rather than vertically as in Figure 1). UP1 in Figure 2 denotes a level 1 UP and UP2 denotes a level 2 UP. BN1 and BN2 are UPBNs of UP1, BN3 and BN4 are UPBNs of UP2. The nodes BN5 and BN6 may be some ToR switches or Servers. The nodes BN3, BN4, BN2, and BN1 are internal to the DC/DCI network (leafs and spines).

```
            ~~~~~~~~~              ~~~~~~~~~
  +-----+  (         )  +-----+  (         )  +-----+
  | BN1 |-(           )-| BN3 |-(           )-| BN5 |
  +-----+ (           ) +-----+ (           ) +-----+
          (    UP1    )         (    UP2    )
  +-----+ (           ) +-----+ (           ) +-----+
  | BN2 |-(           )-| BN4 |-(           )-| BN6 |
  +-----+  (         )  +-----+  (         )  +-----+
            ~~~~~~~~~              ~~~~~~~~~
```

Figure 2 - Example to illustrate partition labels

If the DC network in Figure 2 ran conventional flat distributed BGP-LU control plane using router-allocated labels, when BN5 advertises itself as destination to BN3, BN3 allocates a new label (say L35) from its platform label space. If BN3 finds BN5 reachable (through say LSP35), it advertises L35 (for destination BN5) to BN1. Similarly, BN1 finds BN3 reachable (through say LSP13) and pushes two labels - bottom label is L35 and top label is the LSP13 label. In this model, BN3 stitches L35 to LSP35 that takes the packet to BN5. The same procedure runs on BN4, which allocates a label (say L45, in general different from L35) from its own platform label space for BN5 and advertises the label to BN1. This model is not suitable when end-to-end traffic from a Server behind BN1 or BN2 (not shown in the figure) to a Server behind BN5 or BN6 (not shown in the figure) needs to be forwarded using a label stack imposed by the SDN Controller with the condition that the label stack does not depend on the BN traversed to reach UP2 from UP1.

This document specifies a mechanism to implement the forwarding model

using label stacks imposed by SDN Controller but not have the limitation described in previous paragraph. The new procedures introduced in this document are explained using the above example.

1. BN5 and BN6 advertise their own loopback addresses in UP2. Assuming BN5 and BN6 do not belong to any border group, the BGP-LU advertisements from BN5 and BN6 contain NULL label. The routes will be:
   {Nlri: BN5, Label: NULL, Nh: BN5}
   {Nlri: BN6, Label: NULL, Nh: BN6}

2. BN3 and BN4 do not allocate labels for BN5 and BN6 from their own platform label space when they receive the BGP-LU advertisements. This is because BN3 and BN4 are configured to be part of a border group for UP2 destinations. Both BN3 and BN4 are configured with border group community "Border-group-2".

3. BN3 and BN4 re-advertise BN5 and BN6 as IP NLRI destinations (with BGP next-hop self) to the LMS assigned for UP2 and appends "Partition-Unique Label Info" extended community . The Partition-Unique Label Info extended community and the procedures relating to it are newly introduced in this document. Refer to Section 3.1 for the extended community format and Section 3.2 for LMS procedures. The R-bit in the extended community is set to indicate that the originator requests the receiver to assign and reflect the partition label info community with the label assigned by LMS. The routes for BN5 destination will be:
   {Nlri: BN5, Nh: BN3, Com: Border-group-2, Label-Ext-Comm: R}
   {Nlri: BN5, Nh: BN4, Com: Border-group-2, Label-Ext-Comm: R}

   If the operator has set aside a BGP community value that unambiguously indicates that the next-hop (BN3 or BN4) in the BGP route requests a label to be allocated for the destination (BN5) in UP2 partition, then the newly specified Partition label info extended community may not be added to the route. Refer to Section 3.5 for details.

4. UP2 LMS processes the IP routes for BN5 and BN6, assigns labels for them (or simply reads the labels from label mapping database configured by operator) and originates a BGP-LU route containing the label assigned for the UP2 destinations. LMS may set the P-bit to indicate that the label can be persistent and can be retained for a specified time period. For the two IP routes for BN5 originated by BN3 and BN4, the BGP-LU routes originated by LMS will be:
   {Nlri: BN5, Label: L5, Nh: BN3, Com: Border-group-2, Label-Ext-Comm: P:UP2-context}
   {Nlri: BN5, Label: L5, Nh: BN4, Com: Border-group-2, Label-Ext-

Comm: P:UP2-context}

The procedures if newly specified partition label info extended community is not used are described in Section 3.5.

5. Only when BN3 and BN4 learn the BGP-LU route for BN5 advertised by LMS of UP2, they install the label route in context table corresponding to UP2-context. Note that the operator may configure BN3 and BN4 to install the operator assigned label for BN5 in main LFIB itself (instead of UP2-context). The operator may choose this option if non-overlapping labels are assigned for different UPs.

6. BN3 and BN4 do not advertise BN5 and BN6 in UP1 but only advertise their own loopback addresses. As BN3 and BN4 are configured to be part of a border group, the border group identifier advertised as community is the same in BGP-LU advertisements from BN3 and BN4. If the partitions may have overlapping label spaces, then BN3 and BN4 advertise non-NULL labels in their BGP-LU advertisements. BN3 and BN4 install the label (that gets advertised) in default LFIB and point the label entry to the context table for UP2. In such a case, the routes from BN3 and BN4 will be:
   {Nlri: BN3, Label: CL3, Nh: BN3, Com: Border-group-1}
   {Nlri: BN4, Label: CL4, Nh: BN4, Com: Border-group-1}

7. BN1 and BN2 do not allocate labels for BN3 and BN4 from their platform label space when they receive BGP-LU advertisement. BN1 and BN2 only use the BGP-LU advertisement from BN3 and BN4 for determining the labels to be pushed during forwarding. Note that if there are intermediate routers between BN1/BN2 and BN3/BN4, then the labels CL3 and CL4 advertised by BN3 and BN4 will be used by those intermediate routers for determining the labels to be pushed.

8. BN1 and BN2 re-advertise BN3 and BN4 as IP destinations (with BGP next-hop self) to the LMS assigned for UP2 and appends "Partition-Unique Label Info" extended community. The R-bit is set to indicate that the originator requests the receiver to assign and reflect the partition label info community with the label assigned by LMS. The routes for BN3 destination will be:
   {Nlri: BN3, Nh: BN1, Com: Border-group-1, Label-Ext-Comm: R}
   {Nlri: BN3, Nh: BN2, Com: Border-group-1, Label-Ext-Comm: R}

   The procedures if newly specified partition label info extended community is not used are described in Section 3.5.

9. UP1 LMS processes the IP routes for BN3 and BN4, assigns labels for them (or simply reads the labels from label mapping database configured by operator) and originates a BGP-LU route containing

the label assigned for the UP1 destinations. As the group label advertisements will differ only in BGP next-hop, BGP add-path should be enabled on the peer group between LMS and BNs. LMS may set P-bit to indicate that the advertised label can be persistent and can be retained for specified time. For the two IP routes for BN3 originated by BN1 and BN2, the BGP-LU routes originated by LMS will be:
{Nlri: BN3, Label: L3, Nh: BN1, Com: Border-group-1, Label-Ext-Comm: P:UP1-context}
{Nlri: BN3, Label: LG2, Nh: BN1, Com: Border-group-1, Label-Ext-Comm: PG:UP1-context}
{Nlri: BN3, Label: L3, Nh: BN2, Com: Border-group-1, Label-Ext-Comm: P:UP1-context}
{Nlri: BN3, Label: LG2, Nh: BN2, Com: Border-group-1, Label-Ext-Comm: PG:UP1-context}

Note that there are two BGP-LU routes with same NLRI for advertising group label and so BGP add-path [I-D.ietf-idr-add-paths] should be enabled between LMS and BNs.

10. Only when BN1 and BN2 learn the BGP-LU route for BN3 advertised by LMS of UP1, they install the label route in context table that has been configured on BN1 and BN2 to contain all UP1 destinations. Note that the operator may configure BN1 and BN2 to install the operator assigned label for BN3 in main LFIB itself (instead of UP1-context). The operator may choose this option if non-overlapping labels are assigned for different UPs.

Apart from advertising partition labels to BNs, the LMSs also advertise the routes (IP routes received from BNs as well as the BGP-LU routes originated back to BNs) to Route Resolver. Resolver is logically centralized component that constructs label stacks for end-to-end traffic and it uses the routes advertised from LMSs as inputs for constructing label stacks.

The description of the procedures using the example DC network in Figure 2 provides an overview of how the LFIB states are set up for traffic entering BN1 or BN2 is forwarded to BN5 or BN6 ("downward traffic"). It should be be noted that this overview has not explained how packets from a source in a remote DC can reach BN5 or BN6. In other words, the overview has not yet explained how packets are exchanged between servers in one DC to the other DC in Figure 1. The description of how the LFIB states are setup for "upward traffic" is presented in Section 3.4.

3.1. Partitioned-Unique Label Info Extended Community

This document introduces a new extended community that enables the

originator of a BGP-LU route to convey the information specified below.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type=TBD      | Sub-Type=TBD  | Flags(1 octet)|  Reserved=0   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Partition context identifier                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Partition Label Retention Period                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Flags

R-bit: Set to 1 if the originator requests label

G-bit: Set to 1 if the label is a group label

P-bit: Set to 1 if the receiver can retain label for specified time even if BGP peering between LMS and BN is lost

Partition context identifier: Context table identifier to which label will be installed

Partition label retention period: Timer period in seconds that the label can be retained after the BGP peering between LMS and BN is lost. This value must be zero if P-bit is not set.

3.2 Partition-Unique Label Info Extended Community Procedures

LMS is a BGP speaker that implements the following new procedures when it receives an IP route BGP advertisement containing "Partition-Unique Label Info" extended community.

- If IGP is the routing protocol with in a UP, then LMS may be implemented as a modified Route Reflector (RR) [RFC4456] assigned for the UP.

- If eBGP runs with in a UP, then the BGP peering between LMS and each border node should be configured by operator and on the BNs the eBGP peering with LMS should be configured in a peer group separate from eBGP peering with other routers in the partition. Note that even if eBGP is in use, the LMS procedures may be considered to act as a "modified reflector" because the primary goal of LMS is to return back the partition label to BN.

- LMS is configured with all the border groups that are connected to the UP where each border group is identified by a unique value of

Border-group community.

When LMS receives an IP route advertisement whose NLRI and BGP next-hop are the same, then it executes the following procedure.

1. If the operator has already assigned a label (DstLabel) for the UP destination in the NLRI, then no action is performed.

2. If the operator has not assigned a label for the UP destination, then LMS allocates a label (DstLabel) and stores the mapping between the UP destination and the label.

3. If the IP route advertisement also contains a known Border-group community and if the operator has not assigned a label for the border group, then LMS allocates a label and stores the mapping between the Border-group and the allocated label. Let the label assigned or allocated be BGLabel. LMS also stores the NLRI to the list of nodes belonging to the Border-group community contained in the route.

4. After executing the following procedures, LMS advertises the IP route to the Route Resolver.

When LMS receives an IP route advertisement whose NLRI and BGP next-hop are different, then it executes the following procedures.

1. If the IP route advertisement does not contain "Partition-Unique Label Info" extended community, then no further action is taken. Alternatively, if the LMS is configured with a policy to interpret a BGP community configured on it as equivalent to "partition label info" extended community, then the subsequent steps may be executed (refer to Section 3.5 for details).

2. If the IP route advertisement contains "Partition-Unique Label Info" extended community but the BGP next-hop does not belong to any known Border-group community configured on the LMS, then no further action is taken.

3. If none of the above conditions is true, then the LMS executes the following procedures.

   a. LMS retrieves the DstLabel label already assigned for the UP destination. LMS originates BGP-LU route with DstLabel set in the NLRI and clears the G-bit in "Partition-Unique Label Info" extended community. If the partition labels are operator assigned and is read from label mapping database, then LMS sets P-bit in the extended community flags and sets the "partition label retention period" to the value configured on LMS (default

value is 7200 seconds).

b. If the NLRI of the IP route is equal to a known Border-group community configured on the LMS, then the LMS also retrieves the BGLabel assigned for the Border-group. LMS also originates BGP-LU route with BGLabel set in the NLRI and sets the G-bit in "Partition-Unique Label Info" extended community. If the partition labels operator assigned and is read from label mapping database, then LMS sets P-bit in the extended community flags and sets the "partition label retention period" to the value configured on LMS (default value is 7200 seconds).

When the BN that originated the IP route receives the BGP-LU route "reflected" back by the LMS, it executes the following procedures.

1. BN first checks whether R-bit is cleared in "Partition-Unique Label Info" extended community. If R-bit has been reset, the label in the NLRI is installed in the context table corresponding to the "partition context identifier" present in the extended community. If "partition context identifier" is zero, then BN installs the label entry in default LFIB.

2. If P-bit is set, then BN should retain the label entry in the designated LFIB (context or default) for the time period specified in "partition label retention period" should the BGP peering with LMS is lost. After BGP peering with LMS is lost, the BN should start "label retention timer" for the labels learnt from the LMS. When the BGP peering is restored, BN should reset the "label retention timer" and re-advertise IP routes corresponding to all UP destinations it had originated before. This procedure ensures that both LMS and BNs exchange all requisite routes before reaching steady state again.

3. If P-bit is not set, then BN should delete the label entry immediately when BGP peering with LMS is lost.

4. BN should delete the label entry from the LFIB when LMS withdraws the BGP-LU route containing the "Partition-Unique Label Info" extended community.

3.3 BGP Policies on UPBNs and LMS

The BGP-LU based control plane mechanism specified in this document assumes the following set of policies be applied on various network nodes in HSDN architecture. The policy configurations required are listed below.

- Each UPBN that connects two UPs are configured with a unique Border-group to advertise membership to "border group" or UPBG. For example, in figure 1 UPBN1-1-1 and UPBN1-1-2 are configured with same Border-group community that uniquely represents the connectivity of the two BNs to UP1-1.

- Depending the routing protocol used with in a UP, each UPBN should either have iBGP or eBGP peering sessions such that all lower level UPBNs or end-devices that are connected to the UP learn each other. For example, the BNs present in UP1-1 in Figure 1 are UPBN1-1-1, UPBN1-1-2, UPBN2-1-1 and UPBN2-1-2 and each of them should learn the loopback address of the other BNs.

- Each UP should have a Label Mapping Server (LMS) that advertises to all the UPBNs the operator assigned partition labels corresponding to each UP destination. Destinations of UPi consists of all individual UPBNi+1 connected to UPi and lower level UPBGs connected to UPi. For example, destinations of UP1-1 (Figure 1) are UPBN2-1-1, UPBN2-1-2 and UPBG2-1, and LMS-1-1 will assign and advertise three labels for UP1-1.

- Each BN in a UP should also have iBGP or eBGP peering session with LMS of the UP. For example, all BNs in UP1-1 should have eBGP peering session with LMS-1-1 if UP1-1 runs eBGP routing protocol.

- UPBNj has a policy to automatically export destinations learnt from UPBNi peer group to UPj peer group (where i=j-1). But UPBNj does not export destinations learnt from UPj peer group to UPBNi peer group. This export policy on UPBNj limits the number of BGP advertisements that any network node in UPi has to process apart from limiting the number of LFIB entries in network nodes.

3.4 BGP-LU Procedures for UP0 Destinations

It should be noted that in the example topology in Figure 2, the BNs attached to UP1 and UP2 have been specified as UP destinations for illustration purposes only. Even a remote destination can be considered as a UP destination as long as the route is leaked into the UP. In HSDN architecture, even though the BNs connected to UP0 are remote for the UPBNs from level 2 down to the leaf level, as long as the normal BGP-LU route leaking policy (specified in Section 3.3) is followed, the LMS of the level 2 (or lower level) UPs will have to assign label for BNs in UP0 (or UP0 destinations). For example, UPBN2-1-1 and UPBN2-1-2 (figure 1) will learn UPBN1-1-1, UPBN1-1-2, UPBN1-2-1 and UPBN1-2-2 because UPBN1-2-1 and UPBN1-2-2 are leaked into UP1-1.

In the DC cloud network specified in Figure 1, the following

procedures are executed to enable packets with top label PL0 reach one of UPBNs connecting to UP0. To obtain end-to-end forwarding using a three label stack in a HSDN network with two levels (i.e. Servers located in UP2-x), the LMS of all UP2-x and UP1-x are set up such that they reflect the same label (i.e. PL0 label) for every UP0 destination (BNs as well as border groups).

1. UPBN1-2-1 and UPBN1-2-2 advertise their own loopback addresses in UP0. As the UPBNs are configured to be part of a border group, the border group community is the same in BGP-LU advertisements. If the partitions may have overlapping label spaces, then UPBN1-2-1 and UPBN1-2-2 advertise non-NULL labels in their BGP-LU advertisements. BN3 and BN4 install the label (that gets advertised) in default LFIB and point the label entry to the context table for UP1-2. In such a case, the routes from BN3 and BN4 will be:
   {Nlri: UPBN1-2-1, Label: CL11, Nh: UPBN1-2-1, Com: Border-group-0}
   {Nlri: UPBN1-2-2, Label: CL12, Nh: UPBN1-2-2, Com: Border-group-0}

2. For UPBN1-1-1 and UPBN1-1-2, the routes to UPBN1-2-1 and UPBN1-2-2 are in same partition (i.e. UP0). The label assigned for UPBN1-2-1, UPBN1-2-2 and UPBG1-2 are the same on LMS-0, LMS-1-1 and LMS-2-1. So all BNs in the left hand side DC in Figure 1 install the same label for UPBN1-2-1, UPBN1-2-2 and UPBG1-2.

Note that as all BNs in the DC cloud install the same label for a UP0 destination, the label range on the implementations of all BNs should have common label space (among different platform label spaces on all BNs) that can be set aside for the UP0 destinations. If this is not possible, then all BNs should be configured with a separate context table for UP0 partition. The BGP-LU procedures involving the "Partition-unique label info" community supports both forms of forwarding.

3.5 Advertising labels without partition label extended community

The procedures specified in Section 3.2 may be executed on LMS and border nodes without using the newly partition label info extended community but using an existing BGP community if all the following conditions are true.

- Each partition has a separate LMS such that border nodes connecting two partitions must have separate BGP peering with LMS of the two partitions.

- Both LMS and BNs are configured with a BGP community and both LMS and BNs interpret that community as an indication from the BGP peer that the procedures specified in Section 3 of this document should be

applied. If LMS receives IP route advertisement whose NLRI and next-hop attribute are different and contains the pre-configured BGP community, then LMS should interpret the update as label request from the BGP peer for the IP destination corresponding to the NLRI. Similarly, when BN receives BGP-LU advertisement for which the BN has originated an IP route and if the BGP-LU advertisement contains the pre-configured BGP community, then BN should interpret the update as partition label advertisement from LMS for the IP destination corresponding to the NLRI.

- BNs are configured with the LFIB to which the label advertised by the LMS should be installed. In this model, LMS cannot advertise the LFIB to which the label forwarding entry should be installed.

- Both LMS and BNs are configured with label retention policy in the event of BGP peering between LMS and BNs were to fail. For example, both LMS and BNs may be configured with label retention period of 7200 seconds so that BNs can retain the LFIB entry for 7200 seconds even if BGP peering with LMS fails.

4. Route Resolution in HSDN Architecture

As a consequence of the procedures described in Section 3, Route Resolver of the network will have the knowledge of the destinations in all UPs and the UPBNs that have advertised those UP destinations. Route Resolver uses this information to construct MPLS label stack to forward the packet to desired destination End-device.

Note that the procedure specified in this Section is only for illustration purpose and hence the implementation of Resolver is free to choose a more optimal mechanism to obtain the same result. The resolution for a given DstServer or End-device IP address works as follows.

1. Resolver should have received all BGP-LU routes of all End-devices from the LMSs of all "leaf" UPs with BGP next-hop specifying the UPBN that serves the UP. The Resolver looks up the given DstServer IP address in the resolution database. If the IP address is not present, then Resolver considers the resolution as having failed.

2. If the DstServer has been advertised by LMS of a UP, then the Resolver obtains the BGP next-hop from the BGP-LU route advertisement. The BGP next-hop is the UPBN of the leaf UP. Note that there may be multiple BGP-LU routes advertising the same DstServer. Assuming the policy is to use ECMP for the traffic, the Resolver picks the BGP-LU advertisement having G-bit set in "Partition-Unique Label Info" extended community and adds the BGLabel to the resulting stack. Assuming the DstServer is located

in second level UP and LG2 is the group label, the stack will be {LG2}.

3. Resolver then looks up the UPBN in the resolution database. If the UPBN IP address is not present, then Resolver considers the resolution as having failed. If there is one or more BGP-LU route with the UPBN as the destination, then the Resolver obtains the BGP next-hop(s). Assuming the policy is to use ECMP for the traffic, the Resolver picks the BGP-LU advertisement having G-bit set in "Partition-Unique Label Info" extended community and adds the BGLabel to the resulting stack. Assuming LG1 is the group label of level 1 UPBG, the stack will be {LG1, LG2}.

4. As the resolution has reached level 1 UPBN (that is a BN in UP0), the Resolver looks up the level 1 UPBN in resolution database. There should be multiple BGP-LU routes with level 1 UPBN as destination. Assuming the policy is to use ECMP for the traffic, the Resolver picks the BGP-LU advertisement having G-bit set in "Partition-Unique Label Info" extended community and adds the BGLabel to the resulting stack. Assuming LG0 is the group label of level 0 BG, the stack will be {LG0, LG1, LG2}. At this point the resolution is considered as successful (refer to Section 3.4) and the Resolver returns the resultant label stack to the querying system.

5. Security Considerations

The procedures defined in the document does not necessitate any security considerations.

6. IANA Considerations

This document defines a new extended community type (see Section 3.1).

7. Acknowledgments

We would like to thank Kaliraj Vairavakkalai and Balaji Rajagopalan for their valuable input and feedback.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, May 2001.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, April 2006.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.

9. Informative References

[I-D.fang-mpls-hsdn-for-hsdc] L. Fang, et. al., "MPLS-Based Hierarchical SDN for Hyper-Scale DC/Cloud", draft-fang-mpls-hsdn-for-hsdc-04 (work in progress), July 2015.

[I-D.ietf-idr-add-paths] D. Walton et al., "Advertisement of Multiple Paths in BGP", draft-ietf-idr-add-paths-13 (work in progress), Dec. 2015.

Authors' Addresses

Luyuan Fang
Microsoft
15590 NE 31st St.
Redmond, WA 98052
Email: lufang@microsoft.com

Deepak Bansal
Microsoft
15590 NE 31st St.
Redmond, WA 98052
Email: dbansal@microsoft.com

Chandra Ramachandran
Juniper Networks
Bangalore, India
Email: csekar@juniper.net

Fabio Chiussi
Seattle, Washington 98116
Email: fabiochiussi@gmail.com

Nabil Bitar
Verizon
40 Sylvan Road
Waltham, MA 02145
Email: nabil.bitar@verizon.com

Yakov Rekhter

Global Routing Operations
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

K. Sriram
D. Montgomery
US NIST
D. McPherson
E. Osterweil
Verisign, Inc.
March 9, 2015

Problem Definition and Classification of BGP Route Leaks
draft-ietf-grow-route-leak-problem-definition-01

Abstract

A systemic vulnerability of the Border Gateway Protocol routing system, known as 'route leaks', has received significant attention in recent years. Frequent incidents that result in significant disruptions to Internet routing are labeled "route leaks", but to date we have lacked a common definition of the term. In this document, we provide a working definition of route leaks, keeping in mind the real occurrences that have received significant attention. Further, we attempt to enumerate (though not exhaustively) different types of route leaks based on observed events on the Internet. We aim to provide a taxonomy that covers several forms of route leaks that have been observed and are of concern to Internet user community as well as the network operator community.

Status of This Memo

Table of Contents

1. Introduction 2
2. Working Definition of Route Leaks 3
3. Classification of Route Leaks Based on Documented Events . . 3
4. Summary 6
5. Security Considerations 7
6. IANA Considerations 7
7. Acknowledgements 7
8. Informative References 7
Authors' Addresses 9

1. Introduction

Frequent incidents [Huston2012][Cowie2013][Cowie2010][Madory][Zmijewski][Paseka][LRL][Khare] that result in significant disruptions to Internet routing are commonly called "route leaks". Examination of the details of some of these incidents reveals that they vary in their form and technical details. Before we can discuss solutions to "the route leak problem" we need a clear, technical definition of the problem and its most common forms. In Section 2, we provide a working definition of route leaks, keeping in view many recent incidents that have received significant attention. Further, in Section 3, we attempt to enumerate (though not exhaustively) different types of route leaks based on observed events on the Internet. We aim to provide a taxonomy that covers several forms of route leaks that have been observed and are of concern to Internet user community as well as the network operator community.

2. Working Definition of Route Leaks

A proposed working definition of route leak is as follows:

A "route leak" is the propagation of routing announcement(s) beyond their intended scope. That is, an AS's announcement of a learned BGP route to another AS is in violation of the intended policies of the receiver, the sender and/or one of the ASes along the preceding AS path. The intended scope is usually defined by a set of local redistribution/filtering policies distributed among the ASes involved. Often, these intended policies are defined in terms of the pair-wise peering business relationship between ASes (e.g., customer, provider, peer). For literature related to AS relationships and routing policies, see [Gao][Gill][Luckie]. For measurements of valley-free violations in Internet routing, see [Giotsas][Wijchers].

The result of a route leak can be redirection of traffic through an unintended path which may enable eavesdropping or traffic analysis, and may or may not result in an overload or black-hole. Route leaks can be accidental or malicious, but most often arise from accidental misconfigurations.

The above definition is not intended to be all encompassing. Perceptions vary widely about what constitutes a route leak. Our aim here is to have a working definition that fits enough observed incidents so that the IETF community has a basis for starting to work on route leak mitigation methods.

3. Classification of Route Leaks Based on Documented Events

As illustrated in Figure 1, a common form of route leak occurs when a multi-homed customer AS (such as AS1 in Figure 1) learns a prefix update from one provider (ISP1) and leaks the update to another provider (ISP2) in violation of intended routing policies, and further the second provider does not detect the leak and propagates the leaked update to its customers, peers, and transit ISPs.

```
                                     /\              /\
                                      \ route-leak(P)/
                                       \ propagated /
                                        \          /
             +------------+    peer    +------------+
      ______| ISP1 (AS2) |----------->|  ISP2 (AS3)|---------->
     /       ------------+  prefix(P) +------------+ route-leak(P)
    | prefix |          \   update      /\        \  propagated
     \  (P)  /           \              /          \
      -------   prefix(P) \            /            \
                   update  \          /              \
                            \        /route-leak(P)  \/
                             \/      /
                          +---------------+
                          | customer(AS1) |
                          +---------------+
```

Figure 1: Illustration of the basic notion of a route leak.

We propose the following taxonomy for classification of route leaks aiming to cover several types of recently observed route leaks, while acknowledging that the list is not meant to be exhaustive. In what follows, we refer to the AS that announces a route that is in violation of the intended policies as the "offending AS".

- o Type 1 "U-Turn with Full Prefix": A multi-homed AS learns a prefix route from one upstream ISP and simply propagates the prefix to another upstream ISP. Neither the prefix nor the AS path in the update is altered. This is similar to a straight forward path-poisoning attack [Kapela-Pilosov], but with full prefix. It should be noted that attacks or leaks of this type are often accidental (i.e. not malicious). The update basically makes a U-turn at the attacker's multi-homed AS. The attack (accidental or deliberate) often succeeds because the second ISP prefers customer announcement over peer announcement of the same prefix. Data packets would reach the legitimate destination albeit via the offending AS, unless they are dropped at the offending AS due to its inability to handle resulting large volumes of traffic.

  * Example incidents: Examples of Type 1 route-leak incidents are (1) the Dodo-Telstra incident in March 2012 [Huston2012], (2) the Moratel-PCCW leak of Google prefixes in November 2012 [Paseka], and (3) the VolumeDrive-Atrato incident in September 2014 [Madory].

- o Type 2 "U-Turn with More Specific Prefix": A multi-homed AS learns a prefix route from one upstream ISP and announces a sub-prefix

(subsumed in the prefix) to another upstream ISP. The AS path in the update is not altered. Update is crafted by the attacker to have a subprefix to maximize the success of the attack while reverse path is kept open by the path poisoning techniques as in [Kapela-Pilosov]. Data packets reach the legitimate destination albeit via the offending AS.

* Example incidents: An example of Type 2 route-leak incident is the demo performed at DEFCON-16 in August 2008 [Kapela-Pilosov]. An attacker who deliberately performs a Type 1 route leak (with full prefix) can just as easily perform a Type 2 route leak (with subprefix) to achieve a greater impact.

o Type 3 "Prefix Hijack with Data Path to Legitimate Origin": A multi-homed AS learns a prefix route from one upstream ISP and announces the prefix to another upstream ISP as if it is being originated by it (i.e. strips the received AS path, and re-originates the prefix). This amounts to straightforward hijacking. However, somehow (not attributable to the use of path poisoning trick by the attacker) a reverse path is present, and data packets reach the legitimate destination albeit via the offending AS. But sometimes the reverse path may not be there, and data packets get dropped following receipt by the offending AS.

* Example incidents: Examples of Type 3 route leak include (1) the China Telecom incident in April 2010 [Hiran][Cowie2010][Labovitz], (2) the Belarusian GlobalOneBel route leak incidents in February-March 2013 and May 2013 [Cowie2013], (3) the Icelandic Opin Kerfi-Simmin route leak incidents in July-August 2013 [Cowie2013], and (4) the Indosat route leak incident in April 2014 [Zmijewski].

o Type 4 "Leak of Internal Prefixes and Accidental Deaggregation": An offending AS simply leaks its internal prefixes to one or more of its transit ASes and/or ISP peers. The leaked internal prefixes are often deaggregated subprefixes (i.e. more specifics) of already announced aggregate prefixes. Further, the AS receiving those leaks fails to filter them. Typically these leaked announcements are due to some transient failures within the AS; they are short-lived, and typically withdrawn quickly following the announcements.

* Example incidents: Leaks of internal prefix-routes occur frequently (e.g. multiple times in a week), and the number of prefixes leaked range from hundreds to thousands per incident. One highly conspicuous and widely disruptive leak of internal prefixes happened recently in August 2014 when AS701 and AS705

leaked about 22,000 more specifics of already announced aggregates [Huston2014][Toonk].

- o Type 5 "Lateral ISP-ISP-ISP Leak": This type of route leak typically occurs when, for example, three sequential ISP peers (e.g. ISP-A, ISP-B and ISP-C) are involved, and ISP-B receives a prefix-route from ISP-A and in turn leaks it to ISP-C. The typical routing policy between laterally (i.e. non-hierarchically) peering ISPs is that they should only propagate to each other their respective customer prefixes.

  - * Example incidents: In [Mauch-nanog][Mauch], route leaks of this type are reported by monitoring updates in the global BGP system and finding three or more very large ISP ASNs in a sequence in a BGP update's AS path. Mauch [Mauch] observes that these are anomalies and potentially route leaks because very large ISPs such as ATT, Sprint, Verizon, and Globalcrossing do not in general buy transit services from each other. However, he also notes that there are exceptions when one very large ISP does indeed buy transit from another very large ISP, and accordingly exceptions are made in his detection algorithm for known cases.

- o Type 6 "Leak of Provider Prefixes to Peer": This type of route leak occurs when an offending AS leaks prefix-routes learned from its provider to a lateral peer.

  - * Example incidents: The incidents reported in [Mauch] include the Type 6 leaks.

- o Type 7 "Leak of Peer Prefixes to Provider": This type of route leak occurs when an offending AS leaks prefix-routes learned from a lateral peer to its (the AS's) own provider. These leaked prefix-routes typically originate from the customer cone of the lateral peer.

  - * Example incidents: Some of the example incidents cited for Type 1 route leaks above are also inclusive of Type 7 route leaks. For instance, in the Dodo-Telstra incident [Huston2012], the leaked routes from Dodo to Telstra included routes that Dodo learned from its providers as well as lateral peers.

4. Summary

We attempted to provide a working definition of route leak. We also presented a taxonomy for categorizing route leaks. It covers not all but at least several forms of route leaks that have been observed and are of concern to Internet user and network operator communities. We

hope that this work provides the IETF community a basis for pursuing possible BGP enhancements for route leak detection and mitigation.

5. Security Considerations

No security considerations apply since this is a problem definition document.

6. IANA Considerations

No updates to the registries are suggested by this document.

7. Acknowledgements

The authors wish to thank Danny McPherson and Eric Osterweil for discussions related to this work. Also, thanks are due to Jared Mauch, Jeff Haas, Warren Kumari, Brian Dickson, Amogh Dhamdhere, Jakob Heitz, Geoff Huston, Randy Bush, Ruediger Volk, Andrei Robachevsky, Chris Morrow, and Sandy Murphy for comments, suggestions, and critique. The authors are also thankful to Padma Krishnaswamy, Oliver Borchert, and Okhee Kim for their comments and review.

8. Informative References

[Cowie2010]
Cowie, J., "China's 18 Minute Mystery", Dyn Research/ Renesys Blog, November 2010, <http://research.dyn.com/2010/11/ chinas-18-minute-mystery/>.

[Cowie2013]
Cowie, J., "The New Threat: Targeted Internet Traffic Misdirection", Dyn Research/Renesys Blog, November 2013, <http://research.dyn.com/2013/11/ mitm-internet-hijacking/>.

[Gao] Gao, L. and J. Rexford, "Stable Internet routing without global coordination", IEEE/ACM Transactions on Networking, December 2001, <http://www.cs.princeton.edu/~jrex/papers/ sigmetrics00.long.pdf>.

[Gill] Gill, P., Schapira, M., and S. Goldberg, "A Survey of Interdomain Routing Policies", ACM SIGCOMM Computer Communication Review, January 2014, <https://www.cs.bu.edu/~goldbe/papers/survey.pdf>.

[Giotsas] Giotsas, V. and S. Zhou, "Valley-free violation in Internet routing - Analysis based on BGP Community data", IEEE ICC 2012, June 2012, <http://www0.cs.ucl.ac.uk/staff/V.Giotsas/files/giotsas.icc.2012.pdf>.

[Hiran] Hiran, R., Carlsson, N., and P. Gill, "Characterizing Large-scale Routing Anomalies: A Case Study of the China Telecom Incident", PAM 2013, March 2013, <http://www3.cs.stonybrook.edu/~phillipa/papers/CTelecom.html>.

[Huston2012] Huston, G., "Leaking Routes", March 2012, <http://labs.apnic.net/blabs/?p=139/>.

[Huston2014] Huston, G., "What's so special about 512?", September 2014, <http://labs.apnic.net/blabs/?p=520/>.

[Kapela-Pilosov] Pilosov, A. and T. Kapela, "Stealing the Internet: An Internet-Scale Man in the Middle Attack", DEFCON-16 Las Vegas, NV, USA, August 2008, <https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf/>.

[Khare] Khare, V., Ju, Q., and B. Zhang, "Concurrent Prefix Hijacks: Occurrence and Impacts", IMC 2012, Boston, MA, November 2012, <http://www.cs.arizona.edu/~bzhang/paper/12-imc-hijack.pdf/>.

[LRL] Khare, V., Ju, Q., and B. Zhang, "Large Route Leaks", Project web page, 2012, <http://nrl.cs.arizona.edu/projects/lsrl-events-from-2003-to-2009/>.

[Labovitz] Labovitz, C., "Additional Discussion of the April China BGP Hijack Inciden", Arbor Networks IT Security Blog, November 2010, <http://www.arbornetworks.com/asert/2010/11/additional-discussion-of-the-april-china-bgp-hijack-incident/>.

[Luckie] Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., and kc. claffy, "AS Relationships, Customer Cones, and Validation", IMC 2013, October 2013, <http://www.caida.org/~amogh/papers/asrank-IMC13.pdf>.

[Madory] Madory, D., "Why Far-Flung Parts of the Internet Broke Today", Dyn Research/Renesys Blog, September 2014, <http://research.dyn.com/2014/09/why-the-internet-broke-today/>.

[Mauch] Mauch, J., "BGP Routing Leak Detection System", Project web page, 2014, <http://puck.nether.net/bgp/leakinfo.cgi/>.

[Mauch-nanog] Mauch, J., "Detecting Routing Leaks by Counting", NANOG-41 Albuquerque, NM, USA, October 2007, <https://www.nanog.org/meetings/nanog41/presentations/mauch-lightning.pdf/>.

[Paseka] Paseka, T., "Why Google Went Offline Today and a Bit about How the Internet Works", CloudFare Blog, November 2012, <http://blog.cloudflare.com/why-google-went-offline-today-and-a-bit-about/>.

[Toonk] Toonk, A., "What Caused Today's Internet Hiccup", August 2014, <http://www.bgpmon.net/what-caused-todays-internet-hiccup/>.

[Wijchers] Wijchers, B. and B. Overeinder, "Quantitative Analysis of BGP Route Leaks", RIPE-69, November 2014, <https://ripe69.ripe.net/presentations/157-RIPE-69-Routing-WG.pdf>.

[Zmijewski] Zmijewski, E., "Indonesia Hijacks the World", Dyn Research/Renesys Blog, April 2014, <http://research.dyn.com/2014/04/indonesia-hijacks-world/>.

Authors' Addresses

Kotikalapudi Sriram
US NIST

Email: ksriram@nist.gov

Doug Montgomery
US NIST

Email: dougm@nist.gov

Danny McPherson
Verisign, Inc.

Email: dmcpherson@verisign.com

Eric Osterweil
Verisign, Inc.

Email: eosterweil@verisign.com

Internet Engineering Task Force
Internet-Draft
Updates: 4271 (if approved)
Intended status: Standards Track
Expires: January 7, 2016

W. George
Time Warner Cable
S. Amante
Apple, Inc.
July 6, 2015

Autonomous System Migration Mechanisms and Their Effects on the BGP AS_PATH Attribute
draft-ietf-idr-as-migration-06

Abstract

This draft discusses some existing commonly-used BGP mechanisms for ASN migration that are not formally part of the BGP4 protocol specification. It is necessary to document these de facto standards to ensure that they are properly supported in future BGP protocol work.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
1.2. Documentation note 3
2. ASN Migration Scenario Overview 3
3. External BGP Autonomous System Migration Mechanisms 5
3.1. Modify Inbound BGP AS_PATH Attribute 5
3.2. Modify Outbound BGP AS_PATH Attribute 7
3.3. Implementation 8
4. Internal BGP Autonomous System Migration Mechanisms 9
4.1. Internal BGP AS Migration 10
4.2. Implementation 12
5. Additional Operational Considerations 13
6. IANA Considerations 14
7. Security Considerations 14
8. Acknowledgements 14
9. References 14
9.1. Normative References 14
9.2. Informative References 15
Appendix A. Implementation report 15
Authors' Addresses 16

1. Introduction

This draft discusses some existing commonly-used BGP mechanisms for Autonomous System Number (ASN) migration that are not formally part of the BGP4 [RFC4271] protocol specification. These mechanisms are local to a given BGP Speaker and do not require negotiation with or cooperation of BGP neighbors. The deployment of these mechanisms do not need to interwork with one another to accomplish the desired results, so slight variations between existing vendor implementations exist, and will not necessarily be harmonized due to this document. However, it is necessary to document these de facto standards to ensure that new implementations can be successful, and any future protocol enhancements to BGP that propose to read, copy, manipulate or compare the AS_PATH attribute can do so without inhibiting the use of these very widely used ASN migration mechanisms.

The migration mechanisms discussed here are useful to ISPs and organizations of all sizes, but it is important to understand the business need for these mechanisms and illustrate why they are so critical for ISPs' operations. During a merger, acquisition or divestiture involving two organizations it is necessary to seamlessly migrate both internal and external BGP speakers from one ASN to a

second ASN. The overall goal in doing so is to simplify operations through consistent configurations across all BGP speakers in the combined network. In addition, given that the BGP Path Selection algorithm selects routes with the shortest AS_PATH attribute, it is critical that the ISP does not increase AS_PATH length during or after ASN migration, because an increased AS_PATH length would likely result in sudden, undesirable changes in traffic patterns in the network.

By default, the BGP protocol requires an operator to configure a router to use a single remote ASN for the BGP neighbor, and the ASN must match on both ends of the peering in order to successfully negotiate and establish a BGP session. Prior to the existence of these migration mechanisms, it would have required an ISP to coordinate an ASN change with, in some cases, tens of thousands of customers. In particular, as each router is migrated to the new ASN, to avoid an outage due to ASN mismatch, the ISP would have to force all customers on that router to change their router configurations to use the new ASN immediately after the ASN change. Thus, it becomes critical to allow the ISP to make this process a bit more asymmetric, so that it could seamlessly migrate the ASN within its network(s), but allow the customers to gradually migrate to the ISP's new ASN at their leisure, either by coordinating individual reconfigurations, or accepting sessions using either the old or new ASN to allow for truly asymmetric migration.

1.1. 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 RFC 2119 [RFC2119].

1.2. Documentation note

This draft uses Autonomous System Numbers (ASNs) from the range reserved for documentation as described in RFC 5398 [RFC5398]. In the examples used here, they are intended to represent Globally Unique ASNs, not private use ASNs as documented in RFC 6996 [RFC6996] section 5.

2. ASN Migration Scenario Overview

The use case being discussed here is an ISP merging two or more ASNs, where eventually one ASN subsumes the other(s). In this use case, we will assume the most common case where there are two ISPs, A and B, that prior to the ASN migration use AS 64500 and 64510, respectively. AS 64500 will be the permanently retained ASN used across the consolidated set of both ISPs network equipment, and AS 64510 will be

retired. Thus, at the conclusion of the ASN migration, there will be a single ISP A' with all internal BGP speakers configured to use AS 64500. To all external BGP speakers, the AS_PATH length will not be increased.

In this same scenario, AS 64496 and AS 64499 represent two separate customer networks: C and D, respectively. Originally, customer C (AS 64496) is attached to ISP B, which will undergo ASN migration from AS 64510 to AS 64500. Furthermore, customer D (AS 64499) is attached to ISP A, which does not undergo ASN migration since the ASN for ISP A will remain constant, (AS 64500). Although this example refers to AS 64496 and 64499 as customer networks, either or both may be settlement-free or other types of peers. In this use case they are referred to as "customers" merely for convenience.

```
      ------                  ------
    / ISP  A \              / ISP  B \
   | AS 64500 |            | AS 64510 |
    \        /              \        /
     -------                 -------
        |                       |
        |                       |
  ------------            -------------
  |  Cust D  |            |  Cust C   |
  | AS 64499 |            | AS 64496  |
  ------------            -------------
```

Figure 1: Before Migration

```
              ---------------
            /                 \
           |     ISP A'        |
           |     AS 64500      |
            \                 /
              ---------------
           /                   \
         /                       \
        |                         |
  ------------            -------------
  |  Cust D  |            |  Cust C   |
  | AS 64499 |            | AS 64496  |
  ------------            -------------
```

Figure 2: After Migration

The general order of operations, typically carried out in a single maintenance window by the network undergoing ASN migration (ISP B), are as follows. First, ISP B will change the global BGP ASN used by

a Provider Edge (PE) router, from ASN 64510 to 64500. At this point, the router will no longer be able to establish eBGP sessions toward the existing Customer Edge (CE) devices that are attached to it and still using AS 64510. Second, since ISP B needs to do this without coordinating the simultaneous change of its ASN with all of its eBGP peers, ISP B will configure two separate, but related ASN migration mechanisms discussed in this document on all eBGP sessions toward all CE devices. These mechanisms enable the router to establish BGP neighbors using the legacy ASN, modify the AS_PATH attribute received from a CE device when advertising it further, and modify AS_PATH when transmitted toward CE devices to achieve the desired effect of not increasing the length of the AS_PATH.

At the conclusion of the ASN migration, the CE devices at the edge of the network are not aware of the fact that their upstream router is now in a new ASN and do not observe any change in the length of the AS_PATH attribute. However, after the changes discussed in this document are put in place by ISP A', there is a change to the contents of the AS_PATH attribute to ensure the AS_PATH is not artificially lengthened while these AS migration parameters are used.

In this use case, neither ISP is using BGP Confederations RFC 5065 [RFC5065] internally.

3. External BGP Autonomous System Migration Mechanisms

The following section addresses optional capabilities that are specific to modifying the AS_PATH attribute at the Autonomous System Border Routers (ASBRs) of an organization, (typically a single Service Provider). This ensures that external BGP customers/peers are not forced to make any configuration changes on their CE routers before or during the exact time the Service Provider wishes to migrate to a new, permanently retained ASN. Furthermore, these mechanisms eliminate the artificial lengthening of the AS_PATH both transmitted from and received by the Service Provider that is undergoing AS Migration, which would have negative implications on path selection by external networks.

3.1. Modify Inbound BGP AS_PATH Attribute

The first instrument used in the process described above is called "Local AS". This allows the router to supersede the globally configured ASN in the "My Autonomous System" field of the BGP OPEN [RFC4271] with a locally defined AS value for a specific BGP neighbor or group of neighbors. This mechanism allows the PE router that was formerly in ISP B to establish an eBGP session toward the existing CE devices using the legacy AS, AS 64510. Ultimately, the CE devices (i.e.: customer C) are completely unaware that ISP B has reconfigured

its router to participate as a member of a new AS. Within the context of the former ISP B PE router, the second effect this specific mechanism has on AS_PATH is that, by default, it prepends all received BGP UPDATEs with the legacy AS of ISP B: AS 64510, while advertising it (Adj-RIB-Out) to other BGP speakers (A'). Within the Loc-RIB on ISP B prior to the migration, the AS_PATH of route announcements received from customer C would appear as: 64496, whereas the same RIB on ISP A' (ISP B routers post-migration) would contain AS_PATH: 64510 64496.

A second instrument, referred to as "No Prepend Inbound", is enabled on PE routers migrating from ISP B. The "No Prepend Inbound" capability causes ISP B's routers to not prepend the legacy AS, AS 64510, when advertising UPDATES received from customer C. This restores the AS_PATH within ISP A' for route announcements received from customer C so that it is just one ASN in length: 64496.

In the direction of CE -> PE (inbound):

1. "Local AS": Allows the local BGP router to generate a BGP OPEN to an eBGP neighbor with the old, legacy ASN value in the "My Autonomous System" field. When this capability is activated, it also causes the local router to prepend the <old_ASN> value to the AS_PATH when installing or advertising routes received from a CE to iBGP neighbors inside the Autonomous System.

2. "No Prepend Inbound (of Local AS)": the local BGP router does not prepend <old_ASN> value to the AS_PATH when installing or advertising routes received from the CE to iBGP neighbors inside the Autonomous System

PE-B is a PE that was originally in ISP B, and has a customer eBGP session to CE-B. PE-B has had its global configuration ASN changed from AS 64510 to AS 64500 to make it part of the permanently retained ASN. This now makes PE-B a member of ISP A'. PE-A is a PE that was originally in ISP A, and has a customer peer CE-A. Although its global configuration ASN remains AS 64500, throughout this exercise we also consider PE-A a member of ISP A'.

```
          ISP A'                    ISP A'
     CE-A <--- PE-A <------------------- PE-B <--- CE-B
     64499     New_ASN: 64500   Old_ASN: 64510     64496
                                New_ASN: 64500
```

Note: Direction of BGP UPDATE as per the arrows.

Figure 3: Local AS and No Prepend BGP UPDATE Diagram

As a result using both the "Local AS" and "No Prepend Inbound" capabilities on PE-B, CE-A will see an AS_PATH of: 64500 64496. CE-A will not receive a BGP UPDATE containing AS 64510 in the AS_PATH. (If only the "Local AS" mechanism was configured without "No Prepend Inbound" on PE-B, then CE-A would have seen an AS_PATH of: 64500 64510 64496, which results in an unacceptable lengthening of the AS_PATH). NOTE: If there are still routers in the old ASN (64510), it is possible for them to accept these manipulated routes (i.e. those with 64510 removed from the AS_PATH by this command) as if they have not already passed through their ASN, potentially causing a loop, since BGP's normal loop-prevention behavior of rejecting routes that include its ASN in the path will not catch these. Careful filtering between routers remaining in the old ASN and routers migrated to the new ASN is necessary to minimize the risk of routing loops.

3.2. Modify Outbound BGP AS_PATH Attribute

The two aforementioned mechanisms, "Local AS" and "No Prepend Inbound", only modify the AS_PATH Attribute received by the ISP's PE's in the course of processing BGP UPDATEs from CE devices when CE devices still have an eBGP session established with the ISPs legacy AS (AS64510).

In some existing implementations, "Local AS" and "No Prepend Inbound" does not concurrently modify the AS_PATH Attribute for BGP UPDATEs that are transmitted by the ISP's PE's to CE devices. In these implementations, with "Local AS" and "No Prepend Inbound" used on PE-B, it automatically causes a lengthening of the AS_PATH in outbound BGP UPDATEs from ISP A' toward directly attached eBGP speakers, (Customer C in AS 64496). The externally observed result is that customer C, in AS 64496, will receive the following AS_PATH: 64510 64500 64499. Therefore, if ISP A' takes no further action, it will cause an unacceptable increase in AS_PATH length within customer's networks directly attached to ISP A'.

A tertiary mechanism, referred to as "Replace Old AS", is used to resolve this problem. This capability allows ISP A' to prevent routers from appending the globally configured ASN in outbound BGP UPDATEs toward directly attached eBGP neighbors that are using the "Local AS" mechanism. Instead, only the old (or previously used) AS will be prepended in the outbound BGP UPDATE toward the customer's network, restoring the AS_PATH length to what it what was before AS Migration occurred.

To re-use the above diagram, but in the opposite direction, we have:

```
                    ISP A'                      ISP A'
CE-A ---> PE-A -------------------> PE-B ---> CE-B
64499     New_ASN: 64500   Old_ASN: 64510     64496
                           New_ASN: 64500
```

Note: Direction of BGP UPDATE as per the arrows.

Figure 4: Replace AS BGP UPDATE Diagram

By default, without the use of "Replace Old AS", CE-B would see an AS_PATH of: 64510 64500 64499. After ISP A' changes PE-B to use "Replace Old AS", CE-B would receive an AS_PATH of: 64510 64499, which is the same AS_PATH length pre-AS migration.

3.3. Implementation

The mechanisms introduced in this section MUST be configurable on a per-neighbor or per neighbor group (i.e. a group of similar BGP neighbor statements that reuse some common configuration to simplify provisioning) basis to allow for maximum flexibility. When the "Local AS" capability is used, a local ASN will be provided in the configuration that is different from the globally-configured ASN of the BGP router. To implement this mechanism, a BGP speaker SHOULD send BGP OPEN [RFC4271] (see section 4.2) messages to the configured eBGP peer(s) using the local ASN configured for this session as the value sent in "My Autonomous System". The BGP router SHOULD NOT use the ASN configured globally within the BGP process as the value sent in "My Autonomous System" in the OPEN message. This will avoid causing the eBGP neighbor to unnecessarily generate a BGP OPEN Error message "Bad Peer AS". This method is typically used to re-establish eBGP sessions with peers expecting the legacy ASN after a router has been moved to a new ASN.

Implementations MAY support a more flexible model where the eBGP speaker attempts to open the BGP session using either the ASN configured as "Local AS" or the globally configured AS as discussed in BGP Alias (Section 4.2). If the session is successfully established to the globally configured ASN, then the modifications to AS_PATH described in this document SHOULD NOT be performed, as they are unnecessary. The benefit to this more flexible model is that it allows the remote neighbor to reconfigure to the new ASN without direct coordination between the ISP and the customer.

Note that this procedure will vary slightly if the locally or globally configured ASN is a 4-octet ASN. See section 3 of [RFC4893].

When the BGP router receives UPDATEs from its eBGP neighbor configured with the "Local AS" mechanism, it processes the UPDATE as described in RFC4271 section 5.1.2 [RFC4271]. However the presence of a second ASN due to "Local AS" adds the following behavior to processing UPDATEs received from an eBGP neighbor configured with this mechanism:

1. Internal: the router SHOULD append the configured "Local AS" ASN in the AS_PATH attribute before installing the route or advertising the UPDATE to an iBGP neighbor. The decision of when to append the ASN is an implementation detail outside the scope of this document. Some considerations factoring into this decision include consistency in the AS_PATH throughout the AS, and implementation of the loop detection mechanism.

2. External: the BGP router SHOULD first append the globally configured ASN to the AS_PATH immediately followed by the "Local AS" value before advertising the UPDATE to an eBGP neighbor.

Two options exist to manipulate the behavior of the basic "Local AS" mechanism. They modify the behavior as described below:

1. "No Prepend Inbound" - When the BGP router receives inbound BGP UPDATEs from its eBGP neighbor configured with this option, it MUST NOT append the "Local AS" ASN value in the AS_PATH attribute when installing the route or advertising that UPDATE to iBGP neighbors, but it MUST still append the globally configured ASN as normal when advertising the UPDATE to other local eBGP neighbors (i.e. those natively peering with the globally configured ASN).

2. "Replace Old AS", (outbound) - When the BGP router generates outbound BGP UPDATEs toward an eBGP neighbor configured with this option, the BGP speaker MUST NOT append the globally configured ASN from the AS_PATH attribute. The BGP router MUST append only the configured "Local AS" ASN value to the AS_PATH attribute before sending the BGP UPDATEs outbound to the eBGP neighbor.

4. Internal BGP Autonomous System Migration Mechanisms

The following section describes mechanisms that assist with a gradual and least service impacting migration of Internal BGP sessions from a legacy ASN to the permanently retained ASN. The following mechanism is very valuable to networks undergoing AS migration, but its use does not cause changes to the AS_PATH attribute.

4.1. Internal BGP AS Migration

In this case, all of the routers to be consolidated into a single, permanently retained ASN are under the administrative control of a single entity. Unfortunately, the traditional method of migrating all Internal BGP speakers, particularly within larger networks, is both time consuming and widely service impacting.

The traditional method to migrate Internal BGP sessions was strictly limited to reconfiguration of the global configuration ASN and, concurrently, changing all iBGP neighbors' remote ASN from the legacy ASN to the new, permanently retained ASN on each router within the legacy AS. These changes can be challenging to swiftly execute in networks with with more than a few dozen internal BGP routers. There is also the concomitant service interruptions as these changes are made to routers within the network, resulting in a reset of iBGP sessions and subsequent route reconvergence to reestablish optimal routing paths. Operators often cannot make such sweeping changes given the associated risks of a highly visible service interruption; rather, they require a more gradual method to migrate Internal BGP sessions, from one ASN to a second, permanently retained ASN, that is not visibly service-impacting to its customers.

With the "Internal BGP AS Migration" mechanism described herein, it allows an Internal BGP speaker to form a single iBGP session using either the old, legacy ASN or the new, permanently retained ASN. The benefits of using this mechanism are several fold. First, it allows for a more gradual and less service-impacting migration away from the legacy ASN to the permanently retained ASN. Second, it (temporarily) permits the coexistence of the legacy and permanently retained ASN within a single network, allowing for uniform BGP path selection among all routers within the consolidated network.

The iBGP router with the "Internal BGP AS Migration" capability enabled allows the receipt of a BGP OPEN message with either the legacy ASN value or the new, globally configured ASN value in the "My Autonomous System" field of the BGP OPEN message from iBGP neighbors. It is important to recognize that enablement of the "Internal BGP AS Migration" mechanism preserves the semantics of a regular iBGP session (i.e. using identical ASNs). Thus, the BGP attributes transmitted by and the acceptable methods of operation on BGP attributes received from iBGP sessions configured with "Internal BGP AS Migration" capability are no different than those exchanged across an iBGP session without "Internal BGP AS Migration" configured, as defined by [RFC4271] and [RFC4456].

Typically, in medium to large networks, BGP Route Reflectors [RFC4456] (RRs) are used to aid in reduction of configuration of iBGP

sessions and scalability with respect to overall TCP (and, BGP) session maintenance between adjacent iBGP routers. Furthermore, BGP Route Reflectors are typically deployed in pairs within a single Route Reflection cluster to ensure high reliability of the BGP Control Plane. As such, the following example will use Route Reflectors to aid in understanding the use of the "Internal BGP AS Migration" mechanism. Note that Route Reflectors are not a prerequisite to enable "Internal BGP AS Migration" and this mechanism can be enabled independent of the use of Route Reflectors.

The general order of operations is as follows:

1. Within the legacy network, (the routers comprising the set of devices that still have a globally configured legacy ASN), one member of a redundant pair of RRs has its global configuration ASN changed to the permanently retained ASN. Concurrently, the "Internal BGP AS Migration" capability is enabled on all iBGP sessions on that device. This will comprise Non-Client iBGP sessions to other RRs as well as Client iBGP sessions, typically to PE devices, both still utilizing the legacy ASN. Note that during this step there will be a reset and reconvergence event on all iBGP sessions on the RRs whose configuration was modified; however, this should not be service impacting due to the use of redundant RRs in each RR Cluster.

2. The above step is repeated for the other side of the redundant pair of RRs. The one alteration to the above procedure is that the "Internal BGP AS Migration" mechanism is now removed from the Non-Client iBGP sessions toward the other (previously reconfigured) RRs, since it is no longer needed. The "Internal BGP AS Migration" mechanism is still required on all RRs for all RR Client iBGP sessions. Also during this step, there will be a reset and reconvergence event on all iBGP sessions whose configuration was modified, but this should not be service impacting. At the conclusion of this step, all RRs should now have their globally configured ASN set to the permanently retained ASN and "Internal BGP AS Migration" enabled and in use toward RR Clients.

3. At this point, the network administrators would then be able to establish iBGP sessions between all Route Reflectors in both the legacy and permanently retained networks. This would allow the network to appear to function, both internally and externally, as a single, consolidated network using the permanently retained network.

4. To complete the AS migration, each RR Client (PE) in the legacy network still utilizing the legacy ASN is now modified.

Specifically, each legacy PE would have its globally configured ASN changed to use the permanently retained ASN. The ASN configured within the PE for the iBGP sessions toward each RR would be changed to use the permanently retained ASN. It is unnecessary to enable "Internal BGP AS Migration" mechanism on these migrated iBGP sessions. During the same maintenance window, External BGP sessions would be modified to include the above "Local AS", "No Prepend" and "Replace Old AS" mechanisms described in Section 3 above, since all of the changes are service interrupting to the eBGP sessions of the PE. At this point, all PEs will have been migrated to the permanently retained ASN.

5. The final step is to excise the "Internal BGP AS Migration" configuration from the Router Reflectors in an orderly fashion. After this is complete, all routers in the network will be using the new, permanently retained ASN for all iBGP sessions with no vestiges of the legacy ASN on any iBGP sessions.

The benefit of using the aforementioned "Internal BGP AS Migration" capability is that it is a more gradual and less externally service-impacting change to accomplish an AS migration. Previously, without "Internal BGP AS Migration", such an AS migration change would carry a high risk and need to be successfully accomplished in a very short timeframe (e.g.: at most several hours). In addition, it would likely cause substantial routing churn and rapid fluctuations in traffic carried -- potentially causing periods of congestion and resultant packet loss -- during the period the configuration changes are underway to complete the AS Migration. On the other hand, with "Internal BGP AS Migration", the migration from the legacy ASN to the permanently retained ASN can occur over a period of days or weeks with reduced customer disruption. (The only observable service disruption should be when each PE undergoes the changes discussed in step 4 above.)

4.2. Implementation

The mechanism introduced in this section MUST be configurable on a per-neighbor or per neighbor group basis to allow for maximum flexibility. When configured with this mechanism, a BGP speaker MUST accept BGP OPEN and establish an iBGP session from configured iBGP peers if the ASN value in "My Autonomous System" is either the globally configured ASN or a locally configured ASN provided when this capability is utilized. Additionally, a BGP router configured with this mechanism MUST send its own BGP OPEN [RFC4271] (see section 4.2) using either the globally configured or the locally configured ASN in "My Autonomous System" as follows. To avoid potential deadlocks when two BGP speakers are attempting to establish a BGP

peering session and are both configured with this mechanism, the speaker SHOULD send BGP OPEN using the globally configured ASN first, and only send a BGP OPEN using the locally configured ASN as a fallback if the remote neighbor responds with the BGP error "Bad Peer AS". In each case, the BGP speaker MUST treat UPDATEs sent and received to this peer as if this was a natively configured iBGP session, as defined by [RFC4271] and [RFC4456].

Note that this procedure will vary slightly if the locally or globally configured ASN is a 4-octet ASN. See section 3 of [RFC4893].

5. Additional Operational Considerations

This document describes several mechanisms to support ISPs and other organizations that need to perform ASN migrations. Other variations of these mechanisms may exist, for example, in legacy router software that has not been upgraded or reached End of Life, but continues to operate in the network. Such variations are beyond the scope of this document.

Companies routinely go through periods of mergers, acquisitions and divestitures, which in the case of the former cause them to accumulate several legacy ASNs over time. ISPs often do not have control over the configuration of customers' devices (i.e.: the ISPs are often not providing a managed CE router service, particularly to medium and large customers that require eBGP). Furthermore, ISPs are using methods to perform ASN migration that do not require coordination with customers. Ultimately, this means there is not a finite period of time after which legacy ASNs will be completely expunged from the ISP's network. In fact, it is common that legacy ASNs and the associated External BGP AS Migration mechanisms discussed in this document can and do persist for several years, if not longer. Thus, it is prudent to plan that legacy ASNs and associated External BGP AS Migration mechanisms will persist in a operational network indefinitely.

With respect to the Internal BGP AS Migration mechanism, all of the routers to be consolidated into a single, permanently retained ASN are under the administrative control of a single entity. Thus, completing the migration from iBGP sessions using the legacy ASN to the permanently retained ASN is more straightforward and could be accomplished in a matter of days to months. Finally, good operational hygiene would dictate that it is good practice to avoid using "Internal BGP AS Migration" capability over a long period of time for reasons of not only operational simplicity of the network, but also reduced reliance on that mechanism during the ongoing

lifecycle management of software, features and configurations that are maintained on the network.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

This draft discusses a process by which one ASN is migrated into and subsumed by another. This involves manipulating the AS_PATH Attribute with the intent of not increasing the AS_PATH length, which would typically cause the BGP route to no longer be selected by BGP's Path Selection Algorithm in others' networks. This could result in sudden and unexpected shifts in traffic patterns in the network, potentially resulting in congestion.

Given that these mechanisms can only be enabled through configuration of routers within a single network, standard security measures should be taken to restrict access to the management interface(s) of routers that implement these mechanisms. Additionally, BGP sessions SHOULD be protected using TCP Authentication Option [RFC5925] and the Generalized TTL Security Mechanism [RFC5082]

8. Acknowledgements

Thanks to Kotikalapudi Sriram, Stephane Litkowski, Terry Manderson, David Farmer, Jaroslaw Adam Gralak, Gunter Van de Velde, Juan Alcaide, Jon Mitchell, Thomas Morin, Alia Atlas, Alvaro Retana, and John Scudder for their comments.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, April 2006.

9.2. Informative References

[ALU] Alcatel-Lucent, "BGP Local AS attribute", 2006-2012, <https://infoproducts.alcatel-lucent.com/html/0_add-h-f/93-0074-10-01/7750_SR_OS_Routing_Protocols_Guide/BGP-CLI.html#709567>.

[CISCO] Cisco Systems, Inc., "BGP Support for Dual AS Configuration for Network AS Migrations", 2003, <http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/xe-3s/asr1000/irg-xe-3s-asr1000-book/irg-dual-as.html>.

[JUNIPER] Juniper Networks, Inc., "Configuring the BGP Local Autonomous System Attribute", 2012, <http://www.juniper.net/techpubs/en_US/junos13.3/topics/concept/bgp-local-as-introduction.html>.

[RFC4893] Vohra, Q. and E. Chen, "BGP Support for Four-octet AS Number Space", RFC 4893, May 2007.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, August 2007.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, October 2007.

[RFC5398] Huston, G., "Autonomous System (AS) Number Reservation for Documentation Use", RFC 5398, December 2008.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[RFC6996] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, July 2013.

Appendix A. Implementation report

As noted elsewhere in this document, this set of migration mechanisms has multiple existing implementations in wide use.

- Cisco [CISCO]
- Juniper [JUNIPER]
- Alcatel-Lucent [ALU]

This is not intended to be an exhaustive list, as equivalent features do exist in other implementations, however the authors were unable to find publicly available documentation of the vendor-specific implementation to reference.

Authors' Addresses

Wesley George
Time Warner Cable
13820 Sunrise Valley Drive
Herndon, VA 20171
US

Phone: +1 703-561-2540
Email: wesley.george@twcable.com

Shane Amante
Apple, Inc.
1 Infinite Loop
Cupertino, CA 95014
US

Email: samante@apple.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 9, 2015

Z. Li
Z. Zhuang
Huawei Technologies

March 8, 2015

BGP Extensions for Service-Oriented MPLS Path Programming (MPP)
draft-li-idr-mpls-path-programming-01

Abstract

Service-oriented MPLS programming is to provide customized service process based on flexible label combinations. BGP will play an important role for MPLS path programming to allocate MPLS segment, download programmed MPLS path and the mapping of the service path to the transport path. This document defines BGP extensions to support service-oriented MPLS path programming.

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 RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 2
3. MPLS Segment Allocations 3
4. Download of MPLS Path 3
5. Download of Mapping of Service Path to Transport Path 5
5.1. Extended Unicast Tunnel Attributes 6
5.2. Extended PMSI Tunnel Attribute 7
6. Route Flag Extended Community 9
7. Destination Node Attribute 9
8. Capability Negotiation 10
9. IANA Considerations 11
10. Security Considerations 11
11. References 11
11.1. Normative References 11
11.2. Informative References 12
Authors' Addresses 12

1. Introduction

Service-oriented MPLS programming proposed by [I-D.li-spring-mpls-path-programming] is to provide customized service process based on flexible label combinations. BGP will play an important role for MPLS path programming to allocate MPLS segment, download programmed MPLS path and the mapping of the service path to the transport path. This document defines BGP extensions to support service-oriented MPLS path programming.

2. Terminology

BGP: Border Gateway Protocol

EVPN: Ethernet VPN

L2VPN: Layer 2 VPN

L3VPN: Layer 3 VPN

MPP: MPLS Path Programming

MVPN: Multicast VPN

RR: Route Reflector

SDN: Software-Defined Network

S-EVPN: Segment-based EVPN

SR-Path: Segment Routing Path

NLRI: Network Layer Reachability Information

3. MPLS Segment Allocations

MPLS Segment is the component to compose the MPLS path. [I-D.li-spring-mpls-path-programming] proposes the use cases for service-oriented MPLS path programming which needs following MPLS segments:

1. MPLS path programming for unicast service

o MPLS Segment for VPN identification

o MPLS Segment for ECMP

o MPLS Segment for OAM (Source identification)

o MPLS Segment for Traffic Steering

2. MPLS path programming for multicast service

o MPLS Segment for MVPN identification

o MPLS Segment for Source identification

3. MPLS virtual network for services

o MPLS Segment for MPLS virtual network

These MPLS Segments are defined in individual drafts. It is out of the scope of this document.

4. Download of MPLS Path

According to the service requirements, the central controller can combine MPLS segments flexibly. Then it can download the service label combination for specific prefix related with the service.BGP

extensions are necessary to advertise label stacks for prefix in NLRI field.

```
+---------------------------+
|   Length (1 octet)        |
+---------------------------+
|   Label (3 octets)        |
+---------------------------+
...............................
+---------------------------+
|   Prefix (variable)       |
+---------------------------+
```

Figure 1: NLRI Definition in RFC3107

[RFC3107] defines above NLRI to advertise label binding for specific prefix. The label field can carry one or more labels. Each label is encoded as 3 octets, where the high-order 20 bits contain the label value, and the low order bit contains "Bottom of Stack". But for other AFI/SAFIs using label binding such as VPNv4, VPNv6, EVPN, MVPN, etc., it dose not support the capability to carry more labels for the specific prefix. Moreover for the AFI/SAFIs which do not support label binding capability originally, but may possibly adopt MPLS path programming now, there is no label field in the NLRI. In order to support flexible MPLS path programming, this document defines and uses a new BGP attribute called the "Extended Label attribute". This is an optional transitive BGP attribute. The format of this attribute is defined as follows:

```
+---------------------------+
|   Label 1 (3 octets)      |
+---------------------------+
|   Label 2 (3 octets)      |
+---------------------------+
...............................
+---------------------------+
|   Label n (3 octets)      |
+---------------------------+
```

Figure 2: Extended Label Attribute

The Label field carries one or more labels (that corresponds to the stack of labels [[RFC3032]]). Each label is encoded as 3 octets, where the high-order 20 bits contain the label value, and the low order bit contains "Bottom of Stack" (as defined in [[RFC3032]]).

The Central Controller for MPLS path programming could build a route with Extended Label attribute and send it to the ingress routers.

Upon receiving such a route from the MPP Controller, the ingress router SHOULD select such a route as the best path. If a packet comes into the ingress router and uses such a path, the ingress router will encapsulate the stack of labels which gets from the Extended Label Attribute of the route into the packet and forward the packet along the path.

The "Extended Label attribute" can be used for various BGP address families. Before using this attribute, firstly, it is necessary to negotiate the capability between two nodes to support MPLS path programming for a specific BGP address family. If negotiation fails, a node MUST NOT send this attribute and MUST discard this attribute when it receives.

5. Download of Mapping of Service Path to Transport Path

Since the transport path is also to satisfy the service bearing the requirement, it can also be programmed according to traffic engineering requirements of service. Or the transport path can be set up according to general traffic engineering requirements. Then there needs to be implements the mapping of the service path to the transport path. BGP Extensions is necessary that through the community attribute of BGP, the identifier of the transport path can be carried when distribute label stack for a specific prefix.

```
+---------------------------------+
|  Flags (1 octet)                |
+---------------------------------+
|  Tunnel Type (1 octets)         |
+---------------------------------+
|  MPLS Label (3 octets)          |
+---------------------------------+
|  Tunnel Identifier (variable)   |
+---------------------------------+
```

Figure 3: PMSI Tunnel Attribute in RFC6514

[RFC6514] defines the "P-Multicast Service Interface Tunnel (PMSI Tunnel) attribute". It is shown in the above figure. Since the attribute is always for the specific usage in BGP-based MVPN, it can not be applied to all possible use cases of service-oriented MPLS path programming. This document accordingly defines two new types of BGP attribute for both usage of unicast service path and the multicast service path: Extended Unicast Tunnel Attribute and Extended PMSI Tunnel Attribute.

5.1. Extended Unicast Tunnel Attributes

This document defines and uses a new BGP attribute called the "Extended Unicast Tunnel attribute". This is an optional transitive BGP attribute. The format of this attribute is defined as follows:

```
+-----------------------------------------------+
| Flags (1 octet)                               |
+-----------------------------------------------+
| Tunnel Type (1 octets)                        |
+-----------------------------------------------+
| Tunnel Identifier (variable)                  |
+-----------------------------------------------+
| Tunnel Specific Attributes (Variable)(Optional)|
+-----------------------------------------------+
```

This document defines the following flags:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|  reserved   |S|
+-+-+-+-+-+-+-+-+
```

+ Unicast Tunnel Setup Required (S)

If the S flag is not set, the client node is just to map the service path to the corresponding tunnel. If the S flag is set, the client node MUST set up the tunnel according to the tunnel identifier and the tunnel specific attribute firstly. Then it maps the service path to the corresponding tunnel.

The Tunnel Type identifies the type of the tunneling technology used for the unicast service path. The type determines the syntax and semantics of the Tunnel Identifier field. This document defines the following Tunnel Types:

+ 0 - No tunnel information present
+ 1 - RSVP-TE LSP
+ 2 - LDP LSP
+ 3 - GRE Tunnel
+ 4 - MPLS-based Segment Routing Best-effort Path
+ 5 - MPLS-based Segment Routing Traffic Engineering Path

Tunnel Specific Attributes contains the attributes of the tunnel. The field is optional. The value depends on the tunnel type. It will be defined in the future versions.

When the Tunnel Type is set to "No tunnel information present", the Tunnel attribute carries no tunnel information (no Tunnel Identifier). when the type is used, the tunnel used for the service path is determined by the ingress router.

When the Tunnel Type is set to RSVP - Traffic Engineering (RSVP-TE) Label Switched Path (LSP), the Tunnel Identifier is <C-Type, Tunnel Sender Address, Tunnel ID, Tunnel End-point Address> as specified in [RFC3209]. If C-Type = 7, Tunnel Sender Address and Tunnel End-point Address are IPv4 address in 4 octets. If C-Type = 8, Tunnel Sender Address and Tunnel End-point Address are IPv6 address in 16 octets. The other fields in the RSVP-TE LSP Identifier are the same as specified in [RFC3209].

When the Tunnel Type is set to LDP LSP, the Tunnel Identifier is <Ingress Router's IP Address, Address Family, Prefix Length, Prefix> as specified in [RFC5036].

When the Tunnel Type is set to GRE Tunnel, the Tunnel Identifier is <Ingress Router's IP Address, Address Family, Source IP Address, Destination IP Address>.

When the Tunnel Type is set to MPLS-based Segment Routing Best-effort Path, the Tunnel Identifier is <Ingress Router's IP Address, Address Family, Destination Address>. When the ingress router receives a BGP route with MPLS-based Segment Routing Path Tunnel Identifier in the Extended Unicast Tunnel attribute, it will find the best-effort SR-path based on the destination address.

When the Tunnel Type is set to MPLS-based Segment Routing Traffic Engineering Path, the Tunnel Identifier is <C-Type, Tunnel Sender Address, Tunnel ID, Tunnel End-point Address>. If C-Type = 7, Tunnel Sender Address and Tunnel End-point Address are IPv4 address in 4 octets. If C-Type = 8, Tunnel Sender Address and Tunnel End-point Address are IPv6 address in 16 octets. The tunnel identifier is similar as that of RSVP-TE LSP.

5.2. Extended PMSI Tunnel Attribute

This document defines and uses a new BGP attribute called the "Extended PMSI Tunnel attribute". This is an optional transitive BGP attribute. The format of this attribute is defined as follows:

```
+-----------------------------------------------+
| Flags (1 octet)                               |
+-----------------------------------------------+
| Tunnel Type (1 octets)                        |
+-----------------------------------------------+
| Tunnel Identifier (variable)                  |
+-----------------------------------------------+
| Tunnel Specific Attributes (Variable)(Optional)|
+-----------------------------------------------+
```

This document defines the following flags:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|  reserved   |S|
+-+-+-+-+-+-+-+-+
```

+ PMSI Tunnel Setup Required (S)

If the S flag is not set, the client node is just to map the service path to the corresponding tunnel. If the S flag is set, the client node MUST set up the tunnel according to the tunnel identifier and the tunnel specific attribute firstly. Then it maps the service path to the corresponding tunnel.

The Tunnel Type identifies the type of the tunneling technology used for the multicast service path. The type determines the syntax and semantics of the Tunnel Identifier field. This document defines the following Tunnel Types:

+ 0 - No tunnel information present
+ 1 - RSVP-TE P2MP LSP
+ 2 - mLDP P2MP LSP
+ 3 - PIM-SSM Tree
+ 4 - PIM-SM Tree
+ 5 - BIDIR-PIM Tree
+ 6 - Ingress Replication
+ 7 - mLDP MP2MP LSP

Tunnel Identifier: The definition of Tunnel Identifier is the same as those specified in section 5 of [RFC6514].

Tunnel Specific Attributes contains the attributes of the PMSI tunnel. The field is optional. The value depends on the PMSI tunnel type. It will be defined in the future versions.

6. Route Flag Extended Community

This document defines and uses a new BGP Extended Community called as the "Route Flag Extended Community" which Type value is to be assigned by IANA.

The Route Flag Extended Community is used to carry the flag appointed by a BGP route server (e.g., a central controller).

The format of this exteneded community is defined as follows:

```
    0     1     2     3     4     5     6     7
 +-----+-----+-----+-----+-----+-----+-----+-----+
 |    Type   |  Reserved                   |Flag |
 +-----+-----+-----+-----+-----+-----+-----+-----+

 Flag = 0, Treat as normal route
 Flag = 1, Treat as best route
```

When a router receives a BGP route with a Route Flag Extended Community and the Flag set to "1", it SHOULD use the route as the best route when select the route from multiple routes for a specific prefix.

7. Destination Node Attribute

This document defines and uses a new BGP attribute called as the "Destination Node attribute" which Type value is to be assigned by IANA. The Destination Node attribute is an optional non-transitive attribute that can be applied to any address family.

The Destination Node attribute is used to carry a list of node addresses, which are intended to be used to determine the nodes where the route with such attribute SHOULD be considered. If a node receives a BGP route with a Destination Node attribute, it MUST check the node address list. If one address of the list belongs to this node, the route MUST be used in this node. Otherwise the route MUST be ignored silently.

The format of this attribute is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               AFI             |       SAFI    |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
~               Destination Node Address List                   ~
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

AFI: Address Family Identifier (16 bits).

SAFI: Subsequent Address Family Identifier (8 bits).

Reserved: One octet reserved for special flags

Destination Node Address List: The list of IPv4 (AFI=1) or IPv6 (AFI=2) address.

8. Capability Negotiation

It is necessary to negotiate the capability to support MPLS path programming. The MPLS-Path-Programming Capability is a new BGP capability [RFC5492]. The Capability Code for this capability is to be specified by the IANA. The Capability Length field of this capability is variable. The Capability Value field consists of one or more of the following tuples:

```
+-------------------------------------------------+
|  Address Family Identifier (2 octets)           |
+-------------------------------------------------+
|  Subsequent Address Family Identifier (1 octet) |
+-------------------------------------------------+
|  Send/Receive (1 octet)                         |
+-------------------------------------------------+
```

The meaning and use of the fields are as follows:

Address Family Identifier (AFI): This field is the same as the one used in [RFC4760].

Subsequent Address Family Identifier (SAFI): This field is the same as the one used in [RFC4760].

Send/Receive: This field indicates whether the sender is (a) willing to receive programming MPLS paths from its peer (value 1), (b) would

like to send programming MPLS paths to its peer (value 2), or (c) both (value 3) for the <AFI, SAFI>.

9. IANA Considerations

TBD.

10. Security Considerations

TBD.

11. References

11.1. Normative References

[I-D.li-spring-mpls-path-programming]
Li, Z., "Use Cases and Framwork of Service-Oriented MPLS Path Programming (MPP)", draft-li-spring-mpls-path-programming-00 (work in progress), July 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, February 2009.

11.2. Informative References

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, May 2001.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, February 2012.

Authors' Addresses

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 9, 2015

Z. Li
Z. Zhuang
Huawei Technologies

March 8, 2015

Use Cases and Framework of Service-Oriented MPLS Path Programming (MPP)
draft-li-spring-mpls-path-programming-01

Abstract

Source Packet Routing in Networking (SPRING) architecture for unicast traffic has been proposed to cope with the use cases in traffic engineering, fast re-reroute, service chain, etc. It can leverage existing MPLS dataplane without any modification. In fact, the label stack capability in MPLS would have been utilized well to implement flexible path programming to satisfy all kinds of requirements of service bearing. But in the distributed environment, the flexible programming capability is difficult to implement and always confined to reachability. As the introducing of central control in the network, the flexible MPLS programming capability becomes possible owing to two factors: 1. It becomes easier to allocate label for more purposes than reachability; 2. It is easy to calculate the MPLS path in a global network view. Moreover, the MPLS path programming capability can be utilized to satisfy more requirements of service bearing in the service layer which is defined as service-oriented MPLS path programming. This document defines the concept of MPLS path programming, then proposes use cases, architecture and protocol extension requirements in the service layer for the architecture of Source Packet Routing in Networking (SPRING).

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 RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. Programming Capability of MPLS Path 4
  3.1. History Review 4
  3.2. Gap Analysis of Segment Routing 5
4. Use Cases of Service-Oriented MPLS Path Programming 6
  4.1. Use Cases for Unicast Service 6
    4.1.1. Basic Reachability 6
    4.1.2. VPN Identification 6
    4.1.3. ECMP(Equal Cost Multi-Path) 6
    4.1.4. Service OAM 7
    4.1.5. Traffic Steering 7
  4.2. Use Cases of Multicast Service 7
    4.2.1. Basic Reachability 8
    4.2.2. MVPN Identification 8
    4.2.3. Source Identification 8
  4.3. Use Cases of MPLS Virtual Network 8
  4.4. Use cases of More Label Combinations 9
  4.5. Use cases for Centralized Mapping of Service to Tunnels . 9
5. Framework of Service-Oriented MPLS Path Programming 10
  5.1. Central Control for MPLS Path Programming 10
  5.2. BGP-based MPLS Segment Distribution 11
  5.3. MPLS Service Path Programming 11
    5.3.1. Label Combination and Download of MPLS Path 11

5.3.2. Mapping of Service Path to Service Path 12
5.4. Compatibility 12
5.5. Protocol Extensions Requirements 12
5.5.1. BGP 12
5.5.2. I2RS 13
6. IANA Considerations 13
7. Security Considerations 13
8. References 13
8.1. Normative References 13
8.2. Informative References 13
Authors' Addresses 15

1. Introduction

Source Packet Routing in Networking (SPRING) architecture for unicast traffic has been proposed to cope with the use cases in traffic engineering, fast re-reroute, service chain, etc. It can leverage existing MPLS dataplane without any modification. In fact, the label stack capability in MPLS would have been utilized well to implement flexible path programming to satisfy all kinds of requirements of service bearing. But in the distributed environment, the flexible programming capability is difficult to implement and always confined to reachability. As the introducing of central control in the network, the flexible MPLS programming capability becomes possible owing to two factors: 1. It becomes easier to allocate label for more purposes than reachability; 2. It is easy to calculate the MPLS path in a global network view. Moreover, the MPLS path programming capability can be utilized to satisfy more requirements of service bearing in the service layer which is defined as service-oriented MPLS path programming. This document defines the concept of MPLS path programming, then proposes use cases, architecture and protocol extension requirements in the service layer for the architecture of Source Packet Routing in Networking (SPRING).

2. Terminology

BGP: Border Gateway Protocol

BUM: Broadcast, Unknown unicast and Multicast

EVPN: Ethernet VPN

FRR: Fast Re-Route

L2VPN: Layer 2 VPN

L3VPN: Layer 3 VPN

MPP: MPLS Path Programming

MVPN: Multicast VPN

RR: Route Reflector

SDN: Software-Defined Network

SR-path: Segment Routing Path

3. Programming Capability of MPLS Path

MPLS path is composed by label stacks. Since in the label stack the labels in different layers can represent different meaning and the depth of the label stack can be unlimited in theory, it is possible can make up all kinks of MPLS paths based on the combination of labels. If we look on the combination of MPLS labels as programming, it is can be seen that the MPLS path has high programming capability.

3.1. History Review

The solutions based on MPLS label stack have been widely deployed. For example, in the scenario of Options C inter-AS VPN ([RFC4364]), we assume that LDP over TE is used as the transport tunnel and the TE tunnel starts at the ingress PE, following label stack can be composed by the ingress PE for MPLS path to bear VPN service:

```
+----------+---------+---------+---------+
|VPN Prefix|   BGP   |   LDP   | RSVP-TE |
|   Label  |  Label  |  Label  |  Label  |
+----------+---------+---------+---------+
```

If facility FRR ([RFC4090]) is deployed for the MPLS TE tunnel, once the failure happens, additional label will be pushed for the label stack which is shown as follows:

```
+----------+---------+---------+---------+------------+
|VPN Prefix|   BGP   |   LDP   | RSVP-TE | BYPASS FRR |
|   Label  |  Label  |  Label  |  Label  |    Label   |
+----------+---------+---------+---------+------------+
```

The combination of labels in the above label stack is not simpler than the existing segment routing solution which composes the segment routing path through combination of segments. In fact, this is also a use case of source packet routing. But the combination is not as flexible as the segment routing since the combination of labels is always to cope with the reachability issue with limited capability in the distributed environment as follows:

1. Each label in the label stack is always binded with the reachability to a specific prefix. That is, the purpose of the label binding is limited.

2. It is difficult to implement flexible path calculation based on policy or constraints. For example, MPLS TE proposes rich set of traffic engineering attributes for transport. But it needs complex configurations in each ingress node in an unscalable way. That is, the path calculation and composition capability is limited.

As more concepts on MPLS label are proposed such as entropy label, source label, segment routing, etc., the purpose of label binding expands and the combination of labels can become more flexible. MPLS path programming capability becomes more realistic to satisfy more application scenarios.

3.2. Gap Analysis of Segment Routing

Segment Routing ([I-D.filsfils-spring-segment-routing]) is a typical example of MPLS path programming. The segment based on MPLS label is to represent nodes or agencies in the network. Through the collected information of network segments and path calculation based on the service requirement in the central controller, there will be flexible segment routing paths for the usage of traffic engineering. The SR-path can be advertised to the ingress node through PCE extensions. ([I-D.sivabalan-pce-segment-routing]).

Segment routing can implement source packet routing with high flexibility. On the other hand, there are multiple layers for MPLS path to bear services which is shown in the following figure:

```
          +---+                                +---+
 +--+     |   |     +---+    +---+    +---+    |   |     +--+
 |CE|----|PE1|----| P |----| P |----| P |----|PE2|----|CE|
 +--+     |   |     +---+    +---+    +---+    |   |     +--+
          +---+                                +---+

  o--------o--------- Service Layer------------o--------o

            o--------- Network Layer -----------o

            o-------o--------o---------o-------o  Transport Layer

  o-----o   o-----o  o-----o  o-----o  o-----o   o-----o Link Layer
```

Figure 1: Multiple Layers of Service Bearing

Now the segment routing is to provide the source packet routing in the transport layer. We can call this type of source packet routing as Transport-Oriented MPLS path programming. There will be more application scenarios which needs the source packet routing in the service layer and network layer. We call these types of source packet routing as Service-Oriented MPLS path programming.

4. Use Cases of Service-Oriented MPLS Path Programming

4.1. Use Cases for Unicast Service

4.1.1. Basic Reachability

The basic reachability for VPN service is to allocate label to specific prefix including IP address or MAC address. MPLS path is as follows (using L3VPN as the example):

```
+----------+
|VPN Prefix| ---> Transport
|   Label  |        Tunnel
+----------+
```

4.1.2. VPN Identification

There are several use cases which need to indentify the VPN the packet belongs to in the forwarding plane such as the egress PE node protection for VPN ([I-D.zhang-l3vpn-label-sharing]). MPLS Path can be as follows:

```
+----------+----------+
|VPN Prefix|    VPN   | ---> Transport
|   Label  |   Label  |        Tunnel
+----------+----------+
```

4.1.3. ECMP(Equal Cost Multi-Path)

In order to satisfy ECMP to take full advantage of link bandwidth in the network, the entropy label ([RFC6790]) can be encapsulated. MPLS path can be as follows:

```
+----------+----------+----------+
|  Entropy |VPN Prefix|    VPN   | ---> Transport
|   Label  |   Label  |   Label  |        Tunnel
+----------+----------+----------+
```

4.1.4. Service OAM

OAM is an important requirement for the service. The performance metrics should be measured against the Service Level Agreement(SLA) for the user. Now there are relatively complete and mature OAM mechanism for the point-to-point service. But for LDP LSP, owing to the MP2P model it is difficult to identify the flow from a specific PE based on the label. Source label has been proposed as a possible solution([I-D.chen-mpls-source-label]). When the source label is applied, MPLS path can be as follows:

```
+----------+----------+----------+----------+
|  Entropy |VPN Prefix|    VPN   |  Source  | ---> Transport
|   Label  |   Label  |   Label  |   Label  |        Tunnel
+----------+----------+----------+----------+
```

4.1.5. Traffic Steering

Service traffic may span multiple ASes. It is an important use case to steer traffic at ASBR in an AS to specific ASBR in neighboring AS. There are possible solutions for this type of traffic steering:

1. Traffic Steering based on Transport Tunnel

This method looks on the segment between two ASBRs as the extension of the transport tunnel in an AS. It can steer the traffic through the specific path to the neighboring AS.

2. Traffic Steering in Service/Network Layer

This method is to directly encapsulate the service flow with the steering label in the ingress PE before it enters into the transport tunnel. [I-D.filsfils-spring-segment-routing-central-epe] illustrates the application of Segment Routing to solve the Egress Peer Engineering (EPE) requirement. When this method is applied, the MPLS path can be as follows:

```
+----------+----------+----------+----------+----------+
|  Entropy | Steering |VPN Prefix|    VPN   |  Source  | ---> Transport
|   Label  |   Label  |   Label  |   Label  |   Label  |        Tunnel
+----------+----------+----------+----------+----------+
```

4.2. Use Cases of Multicast Service

4.2.1. Basic Reachability

When MPLS multicast tunnel is applied for the multicast service in BGP-based MVPN, VPLS or EVPN, the basic MPLS path can be as follows:

```
+-----------+
| Multicast |--->     Transport
|  Payload  |     Multicast Tunnel
+-----------+
```

4.2.2. MVPN Identification

When multiple MVPNs shares the MPLS multicast tunnel, it is necessary to encapsulate the label to identify specific MVPN([RFC6514]). The MPLS path can be as follows:

```
+-----------+----------+
| Multicast |   MVPN   | --->     Transport
|  Payload  |   Label  |      Multicast Tunnel
+-----------+----------+
```

4.2.3. Source Identification

In order to implement the split horizon or C-MAC learning in the fowarding plane when MPLS multicast is to bear BUM traffic in L2VPN, it is necessary to introduce the label to identify the source of the BUM traffic([I-D.li-l2vpn-segment-evpn]). The MPLS path is as follows:

```
+-----------+----------+----------+
| Multicast |   EVPN   |  Source  | --->     Transport
|  Payload  |   Label  |   Label  |      Multicast Tunnel
+-----------+----------+----------+
```

4.3. Use Cases of MPLS Virtual Network

The framework of MPLS virtual network has been proposed in [I-D.li-mpls-network-virtualization-framework]. When the unicast service or the multicast service enters into the transport tunnel, it may take different MPLS virtual network identified by the MPLS label for the purpose of QoS routing, security or virtual operations. The MPLS path is as follows:

```
+-------------+      +-----------------+
|   Service   | ---> | Virtual Network | ---> Transport
| Label Stack |      |      Label      |       Tunnel
+-------------+      +-----------------+
```

4.4. Use cases of More Label Combinations

Service-oriented MPLS path programming can make full use of flexible combination of MPLS labels to satisfy different requirements for the service flow. Based on the above proposed use cases, MPLS path can be composed adopting part or whole labels for these use cases based on the service requirement. Besides this, more flexible MPLS label combination may be provided:

1. Hierarchical process or multiple repeated process: The label for the same usage can exist in different layers. Or the process identified by the label can exist in multiple nodes along the path. Then the labels for the same usage can be encapsulated several times in the label stack. The encapsulation can be as follows (using SERVICE LABEL to identify the label for the same service process in different layers):

```
+----------+----------+----------+----------+----------+----------+
| SERVICE  |VPN Prefix| SERVICE  |   VPN    | SERVICE  |  Tunnel  |
|  LABEL   |   Label  |  LABEL   |  Label   |  LABEL   |  Label   |
+----------+----------+----------+----------+----------+----------+
```

2. Special-purpose label indicator: Since the label in the service-oriented MPLS programming is for special-purpose process, it may need a special purpose label to indicate the usage of the label followed the special-purpose labels. For example, the ELI(Entropy Label Indicator) is introduced for the entropy label. This may introduce more labels for the combination.

This document is not to define all possible use cases for the service-oriented path programming. The new use cases can be defined in the future independent document.

4.5. Use cases for Centralized Mapping of Service to Tunnels

In the transport layers, there can be multiple tunnels to one specific destination which satisfy different constraints. In the traditional way, the tunnel is set up by the distributed forwarding nodes. As the PCE-initiated LSP setup [I-D.ietf-pce-pce-initiated-lsp]is introduced, the tunnel can be setup in the central controlled way. In order to satisfy the different service requirements, it is necessary to provide the capability to flexibly map the service to different tunnels. Since the central control point has enough information based on the whole network view, it can be an effective way to map the service to the tunnel by the central point and advertise the mapping information to the end-points of the service to guide the mapping in the forwarding node.

5. Framework of Service-Oriented MPLS Path Programming

5.1. Central Control for MPLS Path Programming

Central control plays an important role in MPLS path programming. It can extend the MPLS path programming capability easily. There are two important functionalities for the central control:

1. Central controlled MPLS label allocation: Label can be allocated centrally for special usage other than reachability. These labels can be used to compose MPLS path. We call it as MPLS Segment.

2. Central controlled MPLS path programming: Central controller can calculate path in a global network view and implement the MPLS path programming based on the collected information of MPLS segments to satisfy different requirements of services.

```
                  +-------------------+
                  |      Central      |
                  |    Controller     |
       |----------|(Path Calculation  |--------|
       |          | /Path Programming)|        |
       |          +-------------------+        |
       |            /      |       \           |
   MPLS Path       /    Segment     \       MPLS Path
       |          /    Allocation    \         |
       |     Segment        |         Segment   |
       |    Allocation      |        Allocation |
       |       /            |            \      |
       |      /             |             \     |
    +--------+         +--------+         +--------+
    | CLIENT |         | CLIENT |         | CLIENT |
    |        | ......  |        | ......  |        |
    |  (PE)  |         |  (P)   |         |  (PE)  |
    |        |         |        |         |        |
    +--------+         +--------+         +--------+
```

Figure 2 Central Control for MPLS Path Programming

There are two types of MPLS path: Transport-Oriented MPLS Path and Service-Oriented MPLS Path. For the transport-oriented MPLS path, segment routing is the typical solution: MPLS segment distribution is done by IGP extensions ([I-D.ietf-isis-segment-routing-extensions]) and [I-D.ietf-ospf-segment-routing-extensions]); the programmed MPLS path can be downloaded through PCEP extensions from PCE to PCC([I-D.sivabalan-pce-segment-routing]). For the service-oriented MPLS path programming, it not only includes composing the MPLS path in the service and network layer, but also includes determining the

mapping of the service path to the transport path. Since the process corresponding to the label in the service label stack is always located at the PE nodes, BGP extensions can be introduced for service-oriented path programming.

5.2. BGP-based MPLS Segment Distribution

1. Label Allocation

There are two types of label used for MPLS segments:

1) Local Label: The service process is done locally. The label can be allocated by the local PE which provides the process.

2) Global Label: The service process is common in multiple PEs. This means the label has global meaning. The label allocation can be done by the central controller. The global label work can refer to [I-D.li-mpls-global-label-framework].

2. Label Mapping Distribution

BGP extensions can be used to distribution label mapping. Regarding to the above two types of label allocation, the process is as follows:

1) Local Label Mapping: BGP can directly distribute the label mapping from the local PE to peer PEs. The local PE can also only distribute the label mapping to central controller. Then the central controller re-distribute the label mapping to other PEs. In this method, the central controller plays the role of traditional RR.

2) Global Label Mapping: The label mapping for the service can be directly distributed by the central controller to multiple PEs. It can be done by BGP extensions.

5.3. MPLS Service Path Programming

5.3.1. Label Combination and Download of MPLS Path

According to the service requirements, the central controller can combine MPLS segments flexibly. Then it can download the service label combination for specific prefix related with the service. The BGP extensions can be reused to download the programmed MPLS path.

5.3.2. Mapping of Service Path to Service Path

Since the transport path is also to satisfy the service bearing the requirement, it can reuse the existing MPLS tunnel technology or it can also be programmed according to traffic engineering requirements of service. Then there needs to be implements the mapping of the service path to the transport path. There are two ways to implement the mapping:

1. BGP Extensions: Through the community attribute of BGP, the identifier of the transport path can be carrier when distribute label stack for a specific prefix.

2. I2RS Extensions: I2RS can be used to download route policy to the client node. Based on the policy, the client node can implement the required mapping.

5.4. Compatibility

When the MPLS path programming is done the central controller and downloaded through BGP extensions to the Client node, the path SHOULD has higher priority than the path calculated on the Client node's own.

5.5. Protocol Extensions Requirements

5.5.1. BGP

REQ 01: BGP extensions SHOULD be introduced to distribute local label mapping for specific process to the central controller and other client nodes.

REQ 02: BGP extensions SHOULD be introduced to distribute global label mapping for specific process from the central controller to the client nodes .

REQ 03: BGP extensions SHOULD be introduced to download label stack for service-oriented MPLS path.

REQ 04: BGP extensions SHOULD be introduced to carry the identifier of the transport MPLS path with service MPLS path to implement the mapping.

REQ 05: BGP extensions SHOULD be introduced to specify the end-points to accept the prefix advertised by the central controller.

REQ 06: BGP extensions SHOULD be introduced to specify the priority for the prefix with attributes of MPLS path programming advertised by the central controller.

REQ 07: When route selection is done in the client node, the path advertised by the central controller SHOULD has higher priority than the path calculated on the client's own.

5.5.2. I2RS

REQ 11: I2RS clients SHOULD provide interface to I2RS agent to download policy to implement the mapping of the service path to the transport path.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

TBD.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[I-D.chen-mpls-source-label]
Chen, M., Xu, X., Li, Z., Fang, L., and G. Mirsky, "MultiProtocol Label Switching (MPLS) Source Label", draft-chen-mpls-source-label-06 (work in progress), October 2014.

[I-D.filsfils-spring-segment-routing]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., and E. Crabbe, "Segment Routing Architecture", draft-filsfils-spring-segment-routing-04 (work in progress), July 2014.

[I-D.filsfils-spring-segment-routing-central-epe]
Filsfils, C., Previdi, S., Patel, K., Aries, E., shaw@fb.com, s., Ginsburg, D., and D. Afanasiev, "Segment Routing Centralized Egress Peer Engineering", draft-filsfils-spring-segment-routing-central-epe-03 (work in progress), January 2015.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-03 (work in progress), October 2014.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-04 (work in progress), February 2015.

[I-D.ietf-pce-pce-initiated-lsp]
Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model", draft-ietf-pce-pce-initiated-lsp-03 (work in progress), March 2015.

[I-D.li-l2vpn-segment-evpn]
Li, Z., Yong, L., and J. Zhang, "Segment-Based EVPN(S-EVPN)", draft-li-l2vpn-segment-evpn-01 (work in progress), February 2014.

[I-D.li-mpls-global-label-framework]
Li, Z., Zhao, Q., Chen, X., Yang, T., and R. Raszuk, "A Framework of MPLS Global Label", draft-li-mpls-global-label-framework-02 (work in progress), July 2014.

[I-D.li-mpls-global-label-usecases]
Li, Z., Zhao, Q., Yang, T., and R. Raszuk, "Use Cases of MPLS Global Label", draft-li-mpls-global-label-usecases-02 (work in progress), July 2014.

[I-D.li-mpls-network-virtualization-framework]
Li, Z. and M. Li, "Framework of Network Virtualization Based on MPLS Global Label", draft-li-mpls-network-virtualization-framework-00 (work in progress), October 2013.

[I-D.sivabalan-pce-segment-routing]
Sivabalan, S., Medved, J., Filsfils, C., Crabbe, E., Raszuk, R., Lopez, V., and J. Tantsura, "PCEP Extensions for Segment Routing", draft-sivabalan-pce-segment-routing-03 (work in progress), July 2014.

[I-D.zhang-l3vpn-label-sharing]
Zhang, M., Zhou, P., and R. White, "Label Sharing for Fast PE Protection", draft-zhang-l3vpn-label-sharing-02 (work in progress), June 2014.

[RFC4090] Pan, P., Swallow, G., and A. Atlas, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, May 2005.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, February 2012.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

Authors' Addresses

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Interdomain Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 24, 2015

S. Litkowski
Orange Business Service
K. Patel
Cisco Systems
J. Haas
Juniper Networks

March 23, 2015

Timestamp support for BGP paths
draft-litkowski-idr-bgp-timestamp-02

Abstract

BGP is more and more used to transport routing information for critical services. Some BGP updates may be critical to be received as fast as possible : for example, in a layer 3 VPN scenario where a dual-attached site is loosing primary connection, the BGP withdraw message should be propagated as fast as possible to restore the service. The same criticity exists for other address-families like multicast VPNs where "join" messages should also be propagated very fast.

Experience of service providers shows that BGP path propagation time may vary depending on network conditions (especially load of BGP speaker on the path) and too long propagation time are affecting customer service.

It is important for service providers to keep track of BGP updates propagation time to monitor quality of service for the customers. It is also important to be able to identify BGP Speakers that are slowing down the propagation.

This document presents a solution to transport timestamps of a BGP path. The solution is targeted to be used using special identified beacon prefixes that are single-homed.

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

Copyright Notice

Table of Contents

1. Problem statement 3
2. Requirements for monitoring BGP path propagation time 4
  2.1. Architecture 4
  2.2. Measurement accuracy 6
    2.2.1. Clock synchronization 6
    2.2.2. Beacon accuracy 6
  2.3. Churn 6
  2.4. Path propagation complexity 7
3. Proposal 8
4. BGP timestamp attribute 9
5. Processing the BGP timestamp attribute 10
  5.1. Inspection list 10
  5.2. Originating a timestamped route in BGP 11
  5.3. Receiving a timestamped route in BGP 11
  5.4. Sending a timestamped route in BGP 13
    5.4.1. Propagating the BGP Timestamp attribute 13
    5.4.2. Setting the send timestamp 13
  5.5. Limiting churn 14
  5.6. Marking stale entries 15

5.7. Inter-AS considerations 19
5.7.1. Drop option 19
5.7.2. Drop AS option 20
5.7.3. Summary option 21
5.7.4. Propagate option 22
5.8. Retrieving timestamp vector 23
5.9. Handling malformed attribute 23
5.10. Impact on update packing 23
6. Compared to BMP 23
7. Deployment considerations 24
8. Security considerations 25
9. Acknowledgements 25
10. IANA Considerations 25
11. Normative References 25
Authors' Addresses 26

1. Problem statement

```
CE3----PE3               PE4 --- CE4 (Source)
          \             /
           RR3       RR4
              \     /
                RR5
               /    \
            RR1     RR2
           / |         \
          /  |          \
CE1----PE1  PE5          PE2 --- CE2
             |
             CE5
```

Figure 1

The figure 1 describes a typical hierarchical RR design where PEs are meshed to local RRs and local RRs are meshed to more centric RRs. We consider a single multicast VPN between all CEs. CE4 is the source, all others may be receivers. The BGP controlplane also supports some other BGP service like L3VPN service.

We consider an event in L3VPN service leading to RR1 being temporarily overloaded (for example, RR1 is processing massive updates due to a router failure or formatting updates for a route-refresh). In the same timeframe, CE1 wants to join the multicast flow from CE4. PE1 propagates the C-multicast route to RR1, but RR1 fails to propagate the route to RR5 because it is busy processing L3VPN. When RR1 finishes the L3VPN job, it would send the C-multicast route to RR5 and updates would be imported by PE4. The

long time to join the flow may cause CE4 to miss part of the multicast flow.

All BGP implementations are different in term of internal processing within an address family or between address family. The issue described above is just given as an example, and the document does not presume that all implementations are suffering from this exact issue. But whatever the implementation, their always be cases where BGP path propagation could be delayed.

Service providers currently lack of efficient solution to keep track of BGP path propagation time as well as solution to identify the BGP speakers causing issues.

BMP (BGP Monitoring Protocol) may be a solution but as several drawbacks (see Section 6).

2. Requirements for monitoring BGP path propagation time

2.1. Architecture

```
                  ---------            -------
                 /         \         /       \
    RTR_SRC1 ----- |   AS1     | -----  |     AS2   |  ---- RTR_DST1
        |        \         /         \       /           |
      Inject       ---------            ---------           Sink point
       point           |                    |
                       |                    |
                  ---------            -------
                 /         \         /       \
    RTR_DST2 ---- |     AS4     |       |     AS3   |  ---- RTR_SRC2_DST2
        |        \         /         \       /             |
        Sink point   ---------            ---------         Inject/Sink
                                                              point
```

Figure 2

```
                    Single AS
      ------------------------------------------
     /                                          \
    |          RR1 ---------- RR2                 |
    |         /   \               \               |
    | RTR_SRC1     \               RTR_DST2       |
    |    |          \                  |          |
    |   Inject      RR3               Sink point  |
    |     point      |                            |
    |               RTR_DST1                     |
    |                |                            |
     \              Sink point                   /
      ------------------------------------------
```

Figure 3

Figure 2 and Figure 3 describes an interAS and a single AS scenario where a service provider wants to monitor BGP path propagation time from a router to multiple routers. In Figure 2, multiple probing routers are attached to multiple ASes. In Figure 3, all probing routers are in the same AS.

The architecture requires some BGP Speaker to originate some NLRI within the BGP controlplane. In the diagram above, they are identified as "Inject point". In order to provide information about propagation delays, the architecture requires introduction of timestamp information. Architecture also needs to identify BGP Speaker causing high propagation delays. As only, specific advertisement will serve for measurement, the architecture requires BGP Speaker to identify NLRIs that must be timestamped. The architecture also requires some BGP Speaker to serve as sink point where a timestamp vector information can be retrieved. The timestamp vector must contain propagation time information for all BGP Speaker that participated in the BGP path. It is so required that each BGP Speaker along the path to add timestamp information. There may be multiple sink points in the network to perform measurement at different location and also different inject points. An external tool may be connected to Sink Points to retrieve the timestamp information. But this is out of scope of the document.

In case of interAS, for security reason, the architecture MUST support hiding detailed timestamp information to the other AS.

Example of usage :

An external tool should command RTR_SRC to originate a probing BGP NLRI. All the BGP Speakers are configured to measure timestamp for this NLRI. The BGP path would propagate across BGP Speakers. Each BGP Speaker may provide timestamp informations. An external tool

connected to sink points will retrieve timestamp vector information for the NLRI.

2.2. Measurement accuracy

2.2.1. Clock synchronization

For the solution to be accurate, it is mandatory for BGP Speaker to be synchronized. This could be ensured easily within a single AS but in a inter domain scenario, it is hard to ensure that all Speakers are synchronized to a good clock source.

The solution MUST include synchronization information associated with the timestamp in order to be able to compare timestamps between them.

2.2.2. Beacon accuracy

In order to be accurate, an implementation SHOULD :

- o ensure that the timestamped NLRIs are processed with the same priority as non timestamped NLRIs.

- o ensure that the processing of adding timestamp information is as lightweight as possible. If some limitation exists, the vendor SHOULD document them.

Using a unique special prefix advertisement from a single location to evaluate propagation time will not provide a detail view of min/max propagation time values as the user will not know where the path for the prefix may be located in a processing queue. Considering a BGP Speaker handling high churn, the advertisement of the path for the special prefix may have a specific place in the long processing queue of the churn depending on the implementation : it may be first, last or somewhere in the middle.

It is required from user to perform sampling to establish propagation time boundaries based on multiple advertisements. Repeated operations of advertisement then withdraw may help in this. See Section 7 for more details.

2.3. Churn

The target solution MUST NOT create more churn in the BGP controlplane.

2.4. Path propagation complexity

When a NLRI is originated in BGP from a point, a BGP path is created. Nothing ensures that all nodes within the BGP controlplane will receive this BGP path. When a concurrent path already exists from the NLRI, the concurrent path may be prefered by some BGP Speaker leading to hiding of the new path. Moreover, even if the NLRI is originated in BGP from a single point, multiple paths may be created within the BGP controlplane, this is inherent to the BGP meshing in place.

As soon as multiple BGP paths are involved, controlplane convergence may be done in multiple steps in order to find the final best path. This convergence may involve multiple BGP path advertisement (replacing each other) between peers.

The goal of our proposal is not to measure the convergence time but to focus on the path propagation time. In a controlplane convergence involving multiple paths for a NLRI, the solution MUST identify timestamp for the event where the NLRI was seen for the first time on a BGP Speaker.

Example :

```
                  Single AS
     ------------------------------------------
    /                            RTR_SRC2- 10/8 \
   |                            /                |
   |          RR1 ---------- RR2                 |
   |         /   \               \               |
   | RTR_SRC1     \               RTR_DST2       |
   |     |         \                            |
   |   10/8        RR3                            |
   |                |                             |
   |               RTR_DST1                      |
   |                                             |
    \                                           /
     ------------------------------------------
```

Figure 4

In the figure above, consider that the service provider is keep tracking of propagation time for real NLRIs (corresponding to customer routes). All the BGP Speakers in our figure are configured to inspect the NLRI 10/8 which is multihomed. We consider that the network is starting and the NLRI has not been propagated yet.

RTR_SRC1 starts to propagate 10/8 within the BGP controlplane. All BGP Speakers considers the path as best and this path will be propagated within the whole controlplane. Each BGP Speaker would add its timestamp information and RTR_DST1 and RTR_DST2 would be able to record the timestamp vector. In this case, the timestamp vector is quite accurate because it represents an end to end propagation.

Now RTR_SRC2 starts to propagate its own path. RR2 has two paths for 10/8 and will choose the best one, let's consider that RTR_SRC2 path is the best one, RTR_SRC2 path will so be propagated and timestamp vector will be updated. RR1 will also have two paths, and we consider that RR1 prefers RTR_SRC1 path, so RTR_SRC2 path will not be propagated by RR1. In this situation, RTR_DST2 will receive the path from RR2 with accurate timestamp (end to end propagation) but RTR_DST1 will never receive it.

We could also consider a stable network situation, where both paths have been advertised for a long time. A network event may occur (e.g. IGP metric change) that would cause a BGP Speaker within a path vector to change its best path. In Figure 10, an IGP event, may cause RR1 to change its decision and prefers the path originated by RTR_SRC2 as best, the path will be propagated with previous received timestamp information that are no more accurate. RTR_DST1 will receive a BGP timestamp vector containing stale (old) timestamp informations as well as new ones.

3. Proposal

Our proposal is based on tagging NLRI with timestamp values along its BGP path propagation. Each BGP Speaker along the path will add timestamp values, so creating a timestamp vector. An ordered list of timestamps would so be built along the path.

```
   BGP Update      BGP Update       BGP Update      BGP Update
   10.0.0.0/8      10.0.0.0/8       10.0.0.0/8      10.0.0.0/8
   Timestamp:      Timestamp:       Timestamp:      Timestamp:
   R1:T1           R1:T1            R1:T1           R1:T1
                   R2:T2            R2:T2           R2:T2
                                    R3:T3           R3:T3
                                                    R4:T4
 R1 ------------> R2 ------------> R3 ------------> R4 ------------> R5
```

Using this mechanism, we can easily identify if a hop within a path is slowing down the propagation.

We propose to use a new BGP attribute, BGP timestamp attribute to encode timestamps information.

4. BGP timestamp attribute

The BGP timestamp (BGP-TS) Attribute is an optional transitive BGP Path Attribute. The attribute type code is TBD.

The value field of the BGP timestamp attribute is defined as an ordered list of timestamp entries, the first entry being the first timestamp entry added (origin):

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Timestamp #1  (variable)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Timestamp #2  (variable)                       |
...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Timestamp #n  (variable)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The timestamps entries are encoded as follows :

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Receive Timestamp #x                          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Send Timestamp #x                             |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ASN                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|T|   Rsvd      |   SyncType    |   EntryType   |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
|                                                               |
|                        Optional variable field                |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- o Receive timestamp : the time at which the BGP path was received. When originating a path in BGP, the timestamp is the originating time. Expressed in seconds and microseconds since midnight (zero hour), January 1, 1970 (UTC). If zero, the time is unavailable. Precision of the timestamp is implementation- dependent.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Timestamp (seconds)                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Timestamp (microseconds)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- o Send timestamp : the time at which the BGP path was exported to the peer. Expressed in seconds and microseconds since midnight (zero hour), January 1, 1970 (UTC). If zero, the time is unavailable. Precision of the timestamp is implementation-dependent.

- o ASN : AS Number of the local node creating the timestamp entry.

- o Flags :

  - * T : Synchronized, if set, the BGP speaker clock is synchronized to an external system.

- o SyncType : defines the stratum as defined in [RFC5905].

- o EntryType : defines the type of Timestamp entry, the following types are defined :

  - * Type 0 : empty. There is no following variable field. This type is to be used in case of timestamp summarization.

  - * Type 1 : IPv4 address, the following variable field will be 4 bytes long and will contain the IPv4 router ID of the local node.

  - * Type 2 : IPv6 address, the following variable field will be 16 bytes long and will contain the IPv6 router ID of the local node.

  - * Type 3 : Stale Indicator, Stale indicates that previous timestamp entries are old. There is no following variable field. The receive timestamp and send timestamp should be set to zero. The ASN is set to the ASN of the local BGP Speaker.

5. Processing the BGP timestamp attribute

5.1. Inspection list

A BGP Speaker supporting the BGP-TS can decide to timestamp only some specific NLRIs. An inspection list may be configured by the user (filter) to apply timestamping on a specific set of BGP NLRIs. By

default, we suggest that a BGP Speaker supporting BGP-TS SHOULD NOT timestamp any BGP NLRIs.

User of our proposal must be aware that using a complex policy to express inspection list may result in more processing that will influence the end to end propagation time. It is expected that the inspection list policy should be kept as simple as possible.

5.2. Originating a timestamped route in BGP

When a BGP Speaker supporting BGP-TS originates a new path in BGP that matches the inspection list, it MUST add the BGP-TS attribute to the BGP path and MUST set the receive timestamp field to the time the path was originated in BGP. At this time of processing, the send timestamp will be set to 0. If the BGP Speaker is synchronized to an external system when originating the route, the S-bit MUST be set in the attribute and the SyncType MUST be set to the current stratum. As mentioned above, the BGP path of the originated route will have a send timestamp value of zero in the BGP LOC-RIB.

5.3. Receiving a timestamped route in BGP

When a BGP Speaker supporting BGP-TS receives a BGP path that matches the inspection list, the implementation MUST record the current time associated with the received path.

The time recording MUST append before the inbound routing policies.

```
                              Inspection
                                       List
          +------------+      +---+   No match    +------------+
   -->    | Adj-RIB-in | -->  | I | -------------> | Rtg pol in |
          | Peer#1     |      | n |                 | Peers#1    | ----->
          +------------+      | s |     +-------+   |            |
                              | p | --> | AddTS |->|            |
                              | e |     +-------+   +------------+
                              | c |   If match
                              | t |
                              |   |
                              | l |
          +------------+      | i |    No match    +------------+
   -->    | Adj-RIB-in | -->  | s | -------------> | Rtg pol in |
          | Peer#2     |      | t |                 | Peers#2    | ----->
          +------------+      |   |     +-------+   |            |
                              |   | --> | AddTS |->|            |
                              |   |     +-------+   +------------+
                              |   |   If match
                              +---+
```

If the path that matches the inspection list and does not contains a BGP-TS attribute, it MUST add a BGP-TS attribute with a timestamp entry :

- o The receive timestamp MUST be set to the recorded time for this BGP path.

- o If the BGP Speaker is synchronized to an external system when receiving the route, the S-bit MUST be set in the attribute and the SyncType MUST be set to the current stratum.

- o The send timestamp MUST be set to zero.

If the path that matches the inspection list and contains a BGP-TS attribute, it MUST append a new timestamp entry in the existing attribute :

- o The receive timestamp MUST be set to the recorded time for this BGP path.

- o If the BGP Speaker is synchronized to an external system when receiving the route, the S-bit MUST be set in the attribute and the SyncType MUST be set to the current stratum.

- o The send timestamp MUST be set to zero.

The process of adding a timestamp entry or adding BGP-TS attribute SHOULD be as light as possible in order to influence the propagation time as lowest as possible.

When a BGP Speaker supporting BGP-TS receives a BGP path that does not the inspection list and contains a BGP-TS attribute, it MUST NOT change the existing attribute.

When a BGP Speaker not supporting BGP-TS receives a BGP path that contains a BGP-TS attribute, it MUST follow the standard BGP procedures described in [RFC4271].

5.4. Sending a timestamped route in BGP

5.4.1. Propagating the BGP Timestamp attribute

For a manageability/security purpose, the authors suggest that BGP timestamp attribute MAY NOT be sent to a peer unless it was explicitly configured for. This would prevent timestamp and internal address informations to be propagated to some external peers for example. See Section 5.7 for more information.

If a BGP path containing a BGP-TS attribute must be sent to be peer not configured with BGP timestamp option, the BGP-TS attribute should be dropped when the update message is sent to the peer.

5.4.2. Setting the send timestamp

If sending timestamp attribute is authorized for a specific peer, and path has a BGP-TS attribute, the outgoing BGP processing MUST fill the send timestamp field when exporting the path to a peer. The time recording MUST occur after all BGP filtering policies (outgoing routing policies, ORF, ...) and after placing path in Adj-RIB-Out. An implementation SHOULD set timestamp at the nearest possible step before sending the BGP Update to the peer. Depending of the implementation, the timestamping may occur at different stage of the outgoing BGP processing. Each implementer SHOULD document their timestamping process in order to make users understand correctly timestamp values. As most of implementations are using the concept of peer-groups, in case, timestamp is set too early in the BGP outgoing processing, all peers within a group may have the same timestamp value. Implementation should avoid this.

The process of adding the send timestamp must be as light as possible in order to influence the propagation time as lowest as possible.

```
+------+
|      |     +--------+     +-----+     +---+   +-------+     No TS
|      | --> | Rtgpol | --> | ORF | --> |...|-->|Adj-RIB|-------------->
|      |     | Out    |     |P#1  |     |   |   |Out    |       Send to peer
|      |     | Peer#1 |     |     |     |   |   |Peer#1 |   +-----+
|      |     |        |     |     |     |   |   |       |-->|AddTS| --->
|      |     +--------+     +-----+     +---+   +-------+   +-----+
|      |                                                     TS present
| BGP  |
| LOC  |
| RIB  |
|      |
|      |     +--------+     +-----+     +---+   +-------+     No TS
|      | --> | Rtgpol | --> | ORF | --> |...|-->|Adj-RIB|-------------->
|      |     | Out    |     |P#2  |     |   |   |Out    |       Send to peer
|      |     | Peer#2 |     |     |     |   |   |Peer#2 |   +-----+
|      |     |        |     |     |     |   |   |       |-->|AddTS| --->
|      |     +--------+     +-----+     +---+   +-------+   +-----+
|      |                                                     TS present
+------+
```

5.5. Limiting churn

Adding timestamp informations to BGP path will make all received paths to be unique.

```
            RR1
          /    \
10/8 - R1        RR3 --- R3
          \    /
            RR2
```

In the figure above, we consider that RR1 and RR2 are part of the same cluster (cluster ID : 1). RR3 is client of RR1 and RR2. R3 is client from RR3, R1 is client from RR1 and RR2.

Without BGP timestamp, when R1 originates the BGP prefix 10/8, it sends it to RR1 and RR2. Consider that RR3 receives path from RR1 first, it will reflect it to R3. When it will receive the path from RR2, it may consider that path from RR2 is best (lowest router ID) but as BGP attributes of the path are exactly the same as for RR1 path, there is no need to send an update to R3.

With BGP timestamp, when R1 originates the BGP prefix 10/8, it sends it to RR1 and RR2. Consider that RR3 receives path from RR1 first,

it will reflect it to R3. When it will receive the path from RR2, it may consider that path from RR2 is best (lowest router ID) but as BGP attributes of the two paths are not more equal due to the timestamp difference, RR3 may need to advertise an update to R3.

In order to prevent introducing more churn, we propose to modify the behavior described in Section 9.2. of [RFC4271]. An implementation MUST NOT consider BGP-TS attribute when evaluating the need to send a new update. As the BGP-TS attribute is purely informational, even if BGP Speakers have a different view of the timestamp attribute, there will be no impact on routing.

Considering our example, when RR3 will receive the path from RR2, even if it considers RR2 path as best, it will not send an update to R3 as all the attributes, except BGP-TS are equal.

5.6. Marking stale entries

Section 2.4 describes some cases where advertised timestamp information is no more relevant because it is old and also requires identification of first propagation timestamps.

In order to do this, we propose to mark old entries by adding a Stale Indicator within the timestamp vector. The presence of Stale Indicator must be interpreted as all previous timestamp entries need to be considered as old and not considered as a first propagation.

BGP-TS attribute example :

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  +
|                Timestamp #1  (IPv4)                           |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Old
|                Timestamp #2   (IPv4)                          |  | entries
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                Timestamp #3   (IPv4)                          |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  +
|                Timestamp #4   (Stale Indicator)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  +
|                Timestamp #5   (IPv4)                          |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Usable
...                                                            ...entries
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                Timestamp #n  (variable)                       |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  +
```

Insertion of Stale Indicator in a BGP-TS attribute may happen in the following conditions :

- o A path is received from a peer containing BGP-TS attribute or originated locally, the path matches the inspection list, and the decision process does not select the path as best path. Then the Stale Indicator SHOULD be inserted after decision process happened.

- o A path is received from a peer containing BGP-TS attribute or originated locally, the path matches the inspection list, and the decision process does select the path as best path. The path is exported to peers and then the Stale Indicator MUST be inserted. The path MUST NOT be repropagated as per Section 5.5.

When inserting a Stale indicator, if a Stale Indicator already exists in the timestamp vector, the implement SHOULD remove it before adding the new one.

```
BGP Update                                                BGP Update
10/8                                                      10/8
NH R2                                                     NH=R1
ASP : 2                                                   ASP : 1,2
Origin IGP                                                Origin IGP
BGP-TS :                                                  BGP-TS :
 [TS_entry1:IPv4]                                          [TS_entry1:IPv4]
 [TS_entry2:IPv4]                                          [TS_entry2:IPv4]
 [TS_entry3:Stale]                                         [TS_entry3:Stale]
 [TS_entry4:IPv4]                                          [TS_entry4:IPv4]
 [TS_entry5:IPv4]                                          [TS_entry5:IPv4]
                                                           [TS_entry6:IPv4]

BGP                        BGP Speaker                            BGP Speaker
Speaker                    R1                                     R3
R2                       +---------------------------+
  ----------------->     |                           | ------------>
                         |   BGP Path                |
                         |   At reception            |
                         | +-----------------------+ |
                         | | 10/8, from R2         | |
                         | |   BGP-TS :            | |
                         | |     [TS_entry1:IPv4]  | |
                         | |     [TS_entry2:IPv4]  | |
                         | |     [TS_entry3:Stale] | |
                         | |     [TS_entry4:IPv4]  | |
                         | |     [TS_entry5:IPv4]  | |
                         | |     [TS_entry6:IPv4]<-| | New timestamp entry
                         | +-----------------------+ | created by R1
                         |                           |
                         |   BGP Path                |
                         |   after sending to peer   |
                         |   Stale state is added    |
                         | +-----------------------+ |
                         | | 10/8, from R2         | |
                         | |   BGP-TS :            | |
                         | |     [TS_entry1:IPv4]  | |
                         | |     [TS_entry2:IPv4]  | |
                         | |     [TS_entry4:IPv4]  | |
                         | |     [TS_entry5:IPv4]  | |
                         | |     [TS_entry6:IPv4]<-| | New timestamp entry
                         | |     [TS_entry7:Stale] | | created by R1
                         | +-----------------------+ |
                         +---------------------------+
```

In the example above, R2 sends a BGP path with some existing stale timestamps. When R1 receives the route, it creates a new timestamp

entry in the BGP-TS attribute. We consider that the decision process decides that the path is best, the path is exported with the new timestamp entry and old timestamps coming from R2. Then R1 will update its local path by removing the previous Stale Indicator and replace a new one at the latest position to mark that it is no more the first propagation.

```
                Single AS
      ----------------------------
     /               RTR_SRC2- 10/8 \
    |              /                 |
    |          RR1                   |
    |         /   \                  |
    | RTR_SRC1     \                 |
    |     |         \                |
    |   10/8        RR3              |
    |                |               |
    |               RTR_DST1         |
     \                              /
      ----------------------------
```

In the figure above, we consider that all BGP Speaker apply timestamp for prefix 10/8. RTR_SRC1 originates 10/8 in BGP, the decision process will decide that the path is best. RTR_SRC1 will export path to RR1 and then it will add locally the Stale Indicator within the timestamp vector. The path exported does not have the Stale Indicator. RR1 will receive the path and add a timestamp entry, the path is considered as best, RR1 will export it to RTR_SRC2 and RR3 and then it will add a stale indicator. RR3 will proceed in the same way.

When RTR_SRC2 will originate a new path for 10/8, if this new path is best on RTR_SRC2, it will export the path to RR1 and then it will add locally the Stale Indicator to the path. When RR1 will receive the route :

- o If the path from RTR_SRC2 is best, RR1 will export the new path to RTR_SRC1 and RR3 and then will add Stale indicator to the path.If RTR_SRC2 fails after some time, RR1 will pick up RTR_SRC1 path as best, and will export it to RR3. RR3 will know that the received timestamp entries are stale thanks to the stale indicator.

- o If the path from RTR_SRC2 is not best, RR1 will add Stale indicator to the path. If RTR_SRC1 fails after some time, RR1 will pick up RTR_SRC2 path as best, and will export it to RR3. RR3 will know that the received timestamp entries are stale thanks to the stale indicator.

5.7. Inter-AS considerations

```
   BGP update
   10.0.0.0/8
   TS:
    AS3;CE1:rT1,sT2


CE1--------->R1 ------------> R2 ------------> R3 ------------> R4 -------> CE2
            |                  |             |                  |
            |                  |             |                  |
        AS3                 AS1                                AS2
  AS4
```

In the figure above, we consider that customer wants to monitor BGP updates propagation time between its two sites.

If AS1 and AS2 BGP Speakers does not support BGP-TS, the attribute will be transported transparently accross AS1 without any processing. CE2 will so receive the BGP path with only a single timestamp entry from CE1.

If AS1 and AS2 BGP Speakers does support BGP-TS, four different options are offered : drop, drop-as, summarize, propagate. It must be noted that using drop-as or summarize options may involve more processing and so may impact the end to end propagation time.

5.7.1. Drop option

If AS1 and/or AS2 BGP Speakers support BGP-TS, they may not want to expose any timestamp information between each other. If a service does not want to propagate timestamp information to external peers, it can decide to not activate the "timestamp" option on the peer configuration , as explained in Section 5.4.

```
   BGP update       BGP update            BGP update        BGP update
   10.0.0.0/8       10.0.0.0/8            10.0.0.0/8        10.0.0.0/8
   TS:              TS:                                     TS:
    AS3;CE1:rT1,sT2  AS3;CE1:rT1,sT2                         AS2;R3:rT5,sT6
                     AS1;R1:rT3,sT4

CE1------------->R1 -----------------> R2 ---------------> R3 ------------> R4
                 |                     | no TS             |                    |
                 |                     |                   |                    |
        AS3                  AS1                                       AS2
```

In the example above, CE1 is configured to send timestamp to R1, as well as R1 to R2. But R2 does not want to send timestamp to R3.

When sending BGP route for 10/8, CE1 adds timestamp attribute and a timestamp entry (AS3, entry type : IPv4=CE1_IP, receive timestamp = T1, send timestamp=T2). R1 receives the path, we suppose that the inspection list matches, so R1 adds a timestamp entry. When sending to R2, R1 will send the following information in its timestamp entry : AS1,entry type : IPv4=R1_IP, receive timestamp T3, send timestamp T4. As R2 is configured to not send timestamp information to R3, it will drop the BGP attribute when sending to R3.

5.7.2. Drop AS option

If AS1 and/or AS2 BGP Speakers support BGP-TS, they may not want to expose their timestamps or internal BGP topology to other ASes. If a service does not want to propagate local AS related timestamp information to external peers, it can decide to use the "drop-as" option towards the peer.

```
   BGP update       BGP update            BGP update         BGP update
   10.0.0.0/8       10.0.0.0/8            10.0.0.0/8         10.0.0.0/8
   TS:              TS:                   TS:                TS:
    AS3;CE1:rT1,sT2  AS3;CE1:rT1,sT2       AS3;CE1:rT1,sT2    AS3;CE1:rT1,sT2
                     AS1;R1:rT3,sT4                           AS2;R3:rT5,sT6

CE1------------->R1 -----------------> R2 ---------------> R3 ------------> R4
                 |                     | no TS             |                    |
                 |                     |                   |                    |
        AS3                  AS1                                       AS2
```

In the example above, CE1 is configured to send timestamp to R1, as well as R1 to R2. But R2 does not want to send AS1 internal timestamp to R3. "Drop-as" option is configured on R2 towards R3.

When sending BGP route for 10/8, CE1 adds timestamp attribute and a timestamp entry (AS3, entry type : IPv4=CE1_IP, receive timestamp = T1, send timestamp=T2). R1 receives the path, we suppose that the inspection list matches, so R1 adds a timestamp entry. When sending to R2, R1 will send the following information in its timestamp entry : AS1,entry type : IPv4=R1_IP, receive timestamp T3, send timestamp T4. As R2 is configured with "drop-as" option to R3, it will remove all timestamp entries where the ASN is equal to its autonomous system number and then send the update to R3.

5.7.3. Summary option

If AS1 and/or AS2 BGP Speakers support BGP-TS, they may want to offer timestamp service to their customers but they want to hide their internal topology. In order to achieve the expected behavior, AS1/ AS2 can activate a timestamp summary option on the external peer.

```
BGP update       BGP update            BGP update          BGP update
10.0.0.0/8       10.0.0.0/8            10.0.0.0/8          10.0.0.0/8
TS:              TS:                   TS:                 TS :
 AS3;CE1:rT1,sT2  AS3;CE1:rT1,sT2       AS3;CE1:rT1,sT2     AS3;CE1:rT1,sT2
                  AS1;R1:rT3,sT4        AS1;rT3,sT5         AS1;rT3,sT5
                                                            AS2;R3,rT6,sT7

CE1------------->R1 -----------------> R2 ---------------> R3 ------------> R4
                 |                     | TS summary        |                |
                 |                     |                   |                |
        AS3                   AS1                                       AS2
```

When using summary option, the BGP-TS attribute is modified as follows when exporting the route :

- o All timestamp entries containing the local AS in AS field are removed.
- o A new timestamp entry is created and inserted in place of removed entries (n entries replaced by 1).
- o The new timestamp entry will use an entry type zero.
- o The new timestamp entry MUST have the S bit set.

- o The new timestamp entry MUST NOT have any EntryType.

- o The receive timestamp of the new timestamp entry is the receiving timestamp of the first timestamp entry that has been removed.

- o The send timestamp of the new timestamp entry will be added as usual.

In the example above, CE1 is configured to send timestamp to R1, as well as R1 to R2. But R2 wants summarize timestamp information to AS2.

When sending BGP route for 10/8, CE1 adds timestamp attribute and a timestamp entry (AS3, entry type : IPv4=CE1_IP, receive timestamp = T1, send timestamp=T2). R1 receives the path, we suppose that the inspection list matches, so R1 adds a timestamp entry. When sending to R2, R1 will send the following information in its timestamp entry : AS1,entry type : IPv4=R1_IP, receive timestamp T3, send timestamp T4. As R2 is configured with "summarize" option to R3, it will remove all timestamp entries where the ASN is equal to its autonomous system number and add a new timestamp entry with an entry type zero. The receive timestamp will be retrieved from R1 timestamp entry.

5.7.4. Propagate option

If AS1 and/or AS2 BGP Speakers support BGP-TS, they may want to offer timestamp service to their customers with a full view. This MUST be the default behavior when timestamp is activated on a peer.

```
BGP update       BGP update            BGP update          BGP update
10.0.0.0/8       10.0.0.0/8            10.0.0.0/8          10.0.0.0/8
TS:              TS:                   TS:                 TS :
 AS3;CE1:rT1,sT2  AS3;CE1:rT1,sT2       AS3;CE1:rT1,sT2     AS3;CE1:rT1,sT2
                  AS1;R1:rT3,sT4        AS1;R1:rT3,sT4      AS1;R1:rT3,sT4
                                        AS1;R2:rT5,sT6      AS1;R2,rT5,sT6
                                                            AS2;R3,rT6,sT7

CE1------------->R1 -----------------> R2 ---------------> R3 ------------> R4
                  |                    | TS propagate      |                |
                  |                    |                   |                |
        AS3                 AS1                                       AS2
```

5.8. Retrieving timestamp vector

Authors suggest to implementers to use a local wrapping buffer on each node and record entries in the buffer each time a BGP path is timestamped. An external tool should then retrieve timestamps information from sink points. How the information is retrieved is out of scope of the document but we can imagine using :

- o BMP from the external tool to the sink point.
- o NetConf get to retrieve wrapping buffer information.
- o SNMP get to retrieve wrapping buffer information.
- o CLI command to retrieve wrapping buffer information.

5.9. Handling malformed attribute

When receiving a BGP Update message containing a malformed BGP-TS attribute, an "attribute discard" action MUST be applied as defined in [I-D.ietf-idr-error-handling].

5.10. Impact on update packing

Introducing timestamps information will make update packing less efficient for the timestamps path. In the deployment we are targeting (Section 7), this is not considered as an issue. In the case where a site is generating a special prefix with path timestamped and others not timestamped, these prefixes will not be packed together, so two update messages will be generated. Even if two updates are generated, we do not consider, that the propagation time will be highly affected.

6. Compared to BMP

BMP (BGP Monitoring Protocol) [I-D.ietf-grow-bmp] is a solution to monitor BGP sessions and provides a convenient interface for obtaining route views. BMP is a complete suite of messages to exchange informations regarding a BGP session.

We can imagine to use BMP as a solution to monitor BGP update propagation time but there is multiple drawbacks associated with such solution :

- o BMP provides dump of all received BGP update (per peer). If we are interested only in probing BGP routes, a strong filtering of information may be needed in BMP messages.

- o BMP does not mandate timestamping of messages (as per [I-D.ietf-grow-bmp] Section 5) : "If the implementation is able to provide information about when routes were received, it MAY provide such information in the BMP timestamp field. Otherwise, the BMP timestamp field MUST be set to zero, indicating that time is not available."

- o BMP may provide (if implementation available) timestamps information only for a single router point of view. If we want to retrieve timestamps of all BGP Speakers on a path, a BMP session is required to all BGP speakers. Correlation (based on known design) is also required at the external tool to order timestamps from each BMP session.

- o If BMP provides timestamp information, it does not provide information on how the router clock is synchronized (free run, NTP, GPS ...).

- o BMP only provides Adj-RIB-in view and does not provide outgoing information.

Using BMP to monitor BGP update propagation may complexify the design of the monitor solution. But as mentioned in Section 1, BMP can be used on specific sink routers to retrieve BGP TS vector.

7. Deployment considerations

This solution is not intended to perform timestamp imposition on all BGP prefixes.

The deployment scenario we are targeting is really to monitor some specific single-homed NLRIs identified by the service provider (see Section 2 as an example).

These NLRIs may be advertised at some injection point in the network, and timestamp vector will be retrieved at some sink points. As pointed in Section 2.2.2 , multiple samples of measurement will be necessary in order to evaluate the propagation time.

These NLRIs should be single-homed in order to ensure an end to end propagation from injection point to sink point. A coordination between injection and sink points based on an external tool is necessary : once a NLRI to be monitored has been advertised, the tool would retrieve the timestamp vector from the sink point.

Service provider may use real prefixes (used for routing) or special prefixes (standard IP prefix but allocated for beaconing). In case of special prefix used, the tool can at regular interval command the

advertisement and withdrawal of the prefix. The tool must ensure that it has retrieved the timestamp vector before withdrawing the prefix and also wait for convergence after withdrawal before advertising back the prefix.

The inspection list should be kept as small as possible by users in order to not introduce processing overhead and as a consequence slow down propagation.

8. Security considerations

Depending of the implementation and router capacity, adding timestamps to BGP path may consume some router resources. As proposed in Section 5.1, by default a BGP Speaker will not timestamp any path and inspection list should be configured to activate timestamping on a subset of paths. Using this approach, we consider that overhead that may be introduced by timestamping BGP paths is well controlled by operators. An external router cannot force an internal router to timestamp.

Providing detailed timestamps information to other ASes may introduce security issues by exposing internal datas (part of BGP topology, IP addresses, internal performance) to external entities. The proposal we make in Section 5.7 solves this security issue by giving flexibility to operators on the level of information he wants to expose to external peers.

9. Acknowledgements

10. IANA Considerations

IANA shall assign a codepoint for the BGP Timestamp attribute. This codepoint will come from the "BGP Path Attributes" registry.

11. Normative References

[I-D.ietf-grow-bmp]
Scudder, J., Fernando, R., and S. Stuart, "BGP Monitoring Protocol", draft-ietf-grow-bmp-07 (work in progress), October 2012.

[I-D.ietf-idr-error-handling]
Chen, E., Scudder, J., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", draft-ietf-idr-error-handling-18 (work in progress), December 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Keyur Patel
Cisco Systems

Email: keyupate@cisco.com

Jeff Haas
Juniper Networks

Email: jhaas@juniper.net

Routing Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 10, 2016

S. Litkowski
Orange
A. Simpson
Alcatel Lucent
K. Patel
Cisco
J. Haas
Juniper Networks
December 8, 2015

Applying BGP flowspec rules on a specific interface set
draft-litkowski-idr-flowspec-interfaceset-03

Abstract

BGP Flow-spec is an extension to BGP that allows for the dissemination of traffic flow specification rules. The primary application of this extension is DDoS mitigation where the flowspec rules are applied in most cases to all peering routers of the network.

This document will present another use case of BGP Flow-spec where flow specifications are used to maintain some access control lists at network boundary. BGP Flowspec is a very efficient distributing machinery that can help in saving OPEX while deploying/updating ACLs. This new application requires flow specification rules to be applied only on a specific subset of interfaces and in a specific direction.

The current specification of BGP Flow-spec does not detail where the flow specification rules need to be applied.

This document presents a new interface-set flowspec action that will be used in complement of other actions (marking, rate-limiting ...). The purpose of this extension is to inform remote routers on where to apply the flow specification.

This extension can also be used in a DDoS mitigation context where a provider wants to apply the filtering only on specific peers.

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

Copyright Notice

Table of Contents

1. Use case 3
  1.1. Specific filtering for DDoS 3
  1.2. ACL maintenance 4
2. Collaborative filtering and managing filter direction 5
3. Interface specific filtering using BGP flowspec 6
4. Interface-set extended community 7
5. Interaction with permanent traffic actions 8
  5.1. Interaction with interface ACLs 9
  5.2. Interaction with flow collection 10
6. Scaling of per interface rules 10
7. Deployment considerations 11
8. Security Considerations 11
9. Acknowledgements 12
10. IANA Considerations 12

11. References 12
11.1. Normative References 12
11.2. Informative References 13
Authors' Addresses 13

1. Use case

1.1. Specific filtering for DDoS

```
                       -----------------    --- (ebgp) - Peer3 (BW 10G)
                      /                  \/
                     |                   /|
                     |                PE --- (ebgp) - Transit1(BW 4x10G)
 Cust1 --- (ebgp) --- PE                  |
                     |                PE ---- (ebgp) - Peer2 (BW 4*10G)
                     |                   \|
 Cust2 --- (ebgp) --- PE                   |----- (ebgp) - Customer3
                    /|                     |
 Peer1(BW10G)-(ebgp) |                PE --- (ebgp) - Transit2(BW 4x10G)
                     |                     |
                      \                   /
                       ------------------
```

Figure 1

The figure 1 above displays a typical service provider Internet network owing Customers, Peers and Transit. To protect pro actively against some attacks (e.g. DNS, NTP ...), the service provider may want to deploy some rate-limiting of some flows on peers and transit links. But depending on link bandwidth, the provider may want to apply different rate-limiting values.

For 4*10G links peer/transit, it may want to apply a rate-limiting of DNS flows of 1G, while on 10G links, the rate-limiting would be set to 250Mbps. Customer interfaces must not be rate-limited.

BGP Flow-spec infrastructure may already be present on the network, and all PEs may have a BGP session running flowspec address family. The Flowspec infrastructure may be reused by the service provider to implement such rate-limiting in a very quick manner and being able to adjust values in future quickly without having to configure each node one by one. Using the current BGP flowspec specification, it would not be possible to implement different rate limiter on different interfaces of a same router. The flowspec rule is applied to all interfaces in all directions or on some interfaces where flowspec is activated but flowspec rule set would be the same among all interfaces.

Section Section 3 will detail a solution to address this use case using BGP Flowspec.

1.2. ACL maintenance

```
                            ---------------    --- (ebgp) - Cust4_VPN
                           /                \/
    Cust1_INT -- (ebgp) --- PE                /|
                           |              PE ------ (ebgp) - Transit1
    Cust3_VPN -- (ebgp) --- PE                 |
                           |              PE ------ (ebgp) - Peer2
                           |                \|
    Cust2_INT -- (ebgp) --- PE                 |----- (ebgp) - Cust4_INT
                          /|                  |
    Peer1 ------ (ebgp) -- |              PE ------ (ebgp) - Transit2
                           |                 |
                            \                /
                             ----------------
```

Figure 2

The figure 1 above displays a typical service provider multiservice network owing Customers, Peers and Transit for Internet, as well as VPN services. The service provider requires to ensure security of its infrastructure by applying ACLs at network boundary. Maintaining and deploying ACLs on hundreds/thousands of routers is really painful and time consuming and a service provider would be interested to deploy/updates ACLs using BGP Flowspec. In this scenario, depending on the interface type (Internet customer, VPN customer, Peer, Transit ...) the content of the ACL may be different.

We foresee two main cases :

- o Maintaining complete ACLs using flowspec : in this case all the ingress ACL are maintained and deployed using BGPFlowspec. See section Section 8 for more details on security aspects.

- o Requirement of a quick deployment of a new filtering term due to a security alert : new security alerts often requires a fast deployment of new ACL terms. Using traditional CLI and hop by hop provisioning, such deployment takes time and network is unprotected during this time window. Using BGP flowspec to deploy such rule, a service provider can protect its network in few seconds. Then the SP can decide to keep the rule permanently in BGP Flowspec or update its ACL or remove the entry (in case equipments are not vulnerable anymore).

Section Section 3 will detail a solution to address this use case using BGP Flowspec.

2. Collaborative filtering and managing filter direction

[RFC5575] states in Section 5. : "This mechanism is primarily designed to allow an upstream autonomous system to perform inbound filtering in their ingress routers of traffic that a given downstream AS wishes to drop.".

In case of networks collaborating in filtering, there is a use case for performing outbound filtering. Outbound filtering allows to apply traffic action one step before and so may allow to prevent impact like congestions.

```
          ----------------------
        /                        \
        |   Upstream provider     |
        \                        /
          -----------------------
            |                 |
            |P2               |P1
          ----------------------
        /                        \
        |        MyAS             |
        \                        /
          -----------------------
```

Figure 3

In the figure above, MyAS is connected to an upstream provider. If a malicious traffic comes in from the upstream provider, it may congestion P1 or P2 links. If MyAS apply inbound filtering on P1/P2 using BGP Flowspec, the congestion issue will not be solved.

Using collaborative filtering, the upstream provider may propose to MyAS to filter malicious traffic sent to it. We propose to enhance [RFC5575] to make myAS able to send BGP FlowSpec updates (on eBGP sessions) to the upstream provider to request outbound filtering on peering interfaces towards MyAS. When the upstream provider will receive the BGP Flowspec update from MyAS, the BGP flowspec update will contain request for outbound filtering on a specific set of interfaces. The upstream provider will apply automatically the requested filter and congestion will be prevented.

3. Interface specific filtering using BGP flowspec

The use case detailed above requires application of different BGP Flowspec rules on different set of interfaces. The basic specification detailed in [RFC5575] does not address this and does not give any detail on where the FlowSpec filter need to be applied.

We propose to introduce, within BGP Flowspec, an identification of interfaces where a particular filter should apply on. Identification of interfaces within BGP Flowspec will be done through group identifiers. A group identifier marks a set of interfaces sharing a common administrative property. Like a BGP community, the group identifier itself does not have any significance. It is up to the network administrator to associate a particular meaning to a group identifier value (e.g. group ID#1 associated to Internet customer interfaces). The group identifier is a local interface property. Any interface may be associated with one or more group identifiers using manual configuration.

When a filtering rule advertised through BGP Flowspec must be applied only to particular sets of interfaces, the BGP Flowspec BGP update will contain the identifiers associated with the relevant sets of interfaces. In addition to the group identifiers, it will also contain the direction the filtering rule must be applied in (see Section 4).

Configuration of group identifiers associated to interfaces may change over time. An implementation MUST ensure that the filtering rules (learned from BGP Flowspec) applied to a particular interface are always updated when the group identifier mapping is changing.

Considering figure 2, we can imagine the following design :

- Internet customer interfaces are associated with group-identifier 1.

- VPN customer interfaces are associated with group-identifier 2.

- All customer interfaces are associated with group-identifier 3.

- Peer interfaces are associated with group-identifier 4.

- Transit interfaces are associated with group-identifier 5.

- All external provider interfaces are associated with group-identifier 6.

- All interfaces are associated with group-identifier 7.

If the service provider wants to deploy a specific inbound filtering on external provider interfaces only, the provider can send the BGP flow specification using group-identifier 6 and including inbound direction.

There are some cases where nodes are dedicated to specific functions (Internet peering, Internet Edge, VPN Edge, Service Edge ...), in this kind of scenario, there is an interest for a constrained distribution of filtering rules that are using the interface specific filtering. Without the constrained route distribution, all nodes will received all the filters even if they are not interested in those filters. Constrained route distribution of flowspec filters would allow for a more optimized distribution.

4. Interface-set extended community

This document proposes a new BGP Route Target extended community called "flowspec interface-set". This document so expands the definition of the Route Target extended community to allow a new value of high order octet (Type field) to be TBD (in addition to the values specified in [RFC4360]).

In order to ease intra-AS and inter-AS use cases, this document proposes to have a transitive as well as a non transitive version of this extended community.

This new BGP Route Target extended community is encoded as follows :

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type (TBD)   |      0x02     |    Autonomous System Number   :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:     AS Number (cont.)         |O|I|      Group Identifier     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The flags are :

- o O : if set, the flow specification rule MUST be applied in outbound direction to the interface set referenced by the following group-identifier.

- o I : if set, the flow specification rule MUST be applied in input direction to the interface set referenced by the following group-identifier.

Both flags can be set at the same time in the interface-set extended community leading to flow rule to be applied in both directions. An interface-set extended community with both flags set to zero MUST be treated as an error and as consequence, the FlowSpec update MUST be discarded.

The Group Identifier is coded as a 14-bit number (values goes from 0 to 16383).

Multiple instances of the interface-set community may be present in a BGP update. This may appear if the flow rule need to be applied to multiple set of interfaces.

Multiple instances of the community in a BGP update MUST be interpreted as a "OR" operation : if a BGP update contains two interface-set communities with group ID 1 and group ID 2, the filter would need to be installed on interfaces belonging to Group ID 1 or Group ID 2.

As using a Route Target, route distribution of flowspec NLRI with interface-set may be subject to constrained distribution as defined in [RFC4684]. Constrained route distribution for flowspec routes using interface-set requires discussion and will be addressed in a future revision of the document.

5. Interaction with permanent traffic actions

[RFC5575] states that BGP Flowspec is primarily designed to allow upstream AS to perform inbound filtering in their ingress routers. This specification does not precise where this ingress filtering should happen in the packet processing pipe.

This proposal enhances [RFC5575] in order to add action on traffic coming from or going to specific interfaces. Based on this enhancement, some new requirements come to implementations.

An implementation SHOULD apply input actions (I bit set) within the input packet processing pipe. An implementation SHOULD apply output actions (O bit set) within the output packet processing pipe.

As input and output processing pipes may also involve already present static/permanent features that will manipulate the packet, the next sections will try to clarify how the static behaviors should interact will BGP flowspec actions.

5.1. Interaction with interface ACLs

Deploying interface specific filters using BGP FlowSpec (dynamic entries) may interfere with existing permanent interface ACL (static entries). The content of the existing permanent ACL MUST NOT be altered by dynamic entries coming from BGP FlowSpec. Permanent ACLs are using a specific ordering which is not compatible with the ordering of FS rules and misordering of ACL may lead to undesirable behaviour. In order, to keep a deterministic and well known behaviour, an implementation SHOULD process the BGP FlowSpec ACL as follows :

- o In inbound direction, the permanent ACL action is applied first followed by FlowSpec action. This gives the primary action to the permanent ACL as it is done today.

- o In outbound direction, FlowSpec action action is applied first followed by permanent ACL. This gives the final action to the permanent ACL as it is done today.

```
                        Inbound filters                         Outbound filter
s
           ---------      -------      ----------      -------      ---------
          |Permanent| -> |Dynamic| -> |Forwarding| -> |Dynamic| -> |Permanent|
```

In order for a flow to be accepted, the flow must be accepted by the two ACLs and a flow is rejected when one of the ACL rejects it as described in the table below :

Permanent ACL entry action	FlowSpec ACL entry action	Result action
Drop	Drop	Drop
Drop	Accept	Drop
Accept	Drop	Drop
Accept	Accept	Accept

Example :

- o ACL permanent IN :

  - * Entry 1 : permit udp from 10/8 to 11/8 port 53

  - * Entry 2 : permit tcp from 10/8 to 11/8 port 22

* Entry 3 : deny ip from 10/8 to 11/8

- ACL dynamic FlowSpec IN :

  * Entry 1 : deny udp from 10.0.0.1/32 to 11/8 port 53

  * Entry 2 : permit tcp from 10/8 to 11/8 port 80

In the example above :

- a UDP flow from 10.0.0.1 to 11.0.0.2 on port 53 will be rejected because the dynamic ACL rejects it.

- a UDP flow from 10.0.0.2 to 11.0.0.2 on port 53 will be accepted because both ACLs accept it.

- a TCP flow from 10.0.0.2 to 11.0.0.2 on port 80 will be rejected because permanent ACL rejects it.

5.2. Interaction with flow collection

A router may activate flow collection features (used in collaboration with Netflow export). Flow collection can be done at input side or output side. As for ACL, an implementation SHOULD process :

- BGP FS rules after the inbound flow collection : in case of DDoS protection, it is important to keep monitoring of attack flows and so performing action, after collection.

- BGP FS rules before the outbound flow collection : purpose of outbound flow collection is really to track flows that are exiting the interface. BGP FS rules should not interfere in this.

```
        Inbound                                                 Outbound
        Flow          BGP                             BGP       Flow
                collection    FS                            FS          collecti
on
           ---------      -------      ----------      -------      ---------
          |Permanent| -> |Dynamic| -> |Forwarding| -> |Dynamic| -> |Permanent|
```

6. Scaling of per interface rules

Creating rules that are applied on specific interfaces may create forwarding rules that may be harder to share.

An implementation SHOULD take care about trying to keep sharing forwarding structures as much as possible in order to limit the

scaling impact. How the implementation would do so is out of scope of the document.

7. Deployment considerations

There are some cases where a particular BGP Flowspec NLRI may be advertised to different interface groups with a different action. For example, a service provider may want to discard all ICMP traffic from customer interfaces to infrastructure addresses and want to rate-limit the same traffic when it comes from some internal platforms. These particular cases require ADD-PATH to be deployed in order to ensure that all paths (NLRI+interface group+actions) are propagated within the BGP control plane. Without ADD-PATH, only a single "NLRI+interface group+actions" will be propagated, so some filtering rules will never be applied.

8. Security Considerations

Managing permanent Access Control List by using BGP Flowspec as described in Section 1.2 helps in saving roll out time of such ACL. However some ACL especially at network boundary are critical for the network security and loosing the ACL configuration may lead to network open for attackers.

By design, BGP flowspec rules are ephemeral : the flow rule exists in the router while the BGP session is UP and the BGP path for the rule is valid. We can imagine a scenario where a Service Provider is managing the network boundary ACLs by using only FlowSpec. In this scenario, if , for example, an attacker succeed to make the internal BGP session of a router to be down , it can open all boundary ACLs on the node, as flowspec rules will disappear due to the BGP session down.

In reality, the chance for such attack to occur is low, as boundary ACLs should protect the BGP session from being attacked.

In order to complement the BGP flowspec solution is such deployment scenario and provides security against such attack, a service provider may activate Long lived Graceful Restart [I-D.uttaro-idr-bgp-persistence] on the BGP session owning Flowspec address family. So in case of BGP session to be down, the BGP paths of Flowspec rules would be retained and the flowspec action will be retained.

9. Acknowledgements

Authors would like to thanks Wim Hendrickx for his valuable comments.

10. IANA Considerations

This document requests a new type from the "BGP Transitive Extended Community Types" extended community registry. This type name shall be 'FlowSpec'.

This document requests a new type from the "BGP Non-Transitive Extended Community Types" extended community registry. This type name shall be 'FlowSpec'.

This document requests creation of a new registry called "FlowSpec Extended Community Sub-Types". This registry contains values of the second octet (the "Sub-Type" field) of an extended community when the value of the first octet (the "Type" field) is to one of those allocated in this document.

Within this new registry, this document requests a new subtype (suggested value 0x02), this sub-type shall be named "interface-set".

11. References

11.1. Normative References

[I-D.ietf-idr-rtc-no-rt]
Rosen, E., Patel, K., Haas, J., and R. Raszuk, "Route Target Constrained Distribution of Routes with no Route Targets", draft-ietf-idr-rtc-no-rt-04 (work in progress), November 2015.

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

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <http://www.rfc-editor.org/info/rfc4360>.

[RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <http://www.rfc-editor.org/info/rfc4684>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <http://www.rfc-editor.org/info/rfc5575>.

11.2. Informative References

[I-D.uttaro-idr-bgp-persistence]
Uttaro, J., Chen, E., Decraene, B., and J. Scudder, "Support for Long-lived BGP Graceful Restart", draft-uttaro-idr-bgp-persistence-03 (work in progress), November 2013.

Authors' Addresses

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Adam Simpson
Alcatel Lucent

Email: adam.simpson@alcatel-lucent.com

Keyur Patel
Cisco

Email: keyupate@cisco.com

Jeff Haas
Juniper Networks

Email: jhaas@juniper.net

Interdomain Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

S. Litkowski
Orange Business Service
J. Haas
Juniper Networks
K. Patel
Cisco Systems
March 5, 2015

Inter Domain considerations for Constrained Route distribution
draft-litkowski-idr-rtc-interas-01

Abstract

[RFC4684] defines Multi-Protocol BGP (MP-BGP) procedures that allow BGP speakers to exchange Route Target reachability information in order to limit the propagation of Virtual Private Networks (VPN) Network Layer Reachability Information (NLRI).

[RFC4684] addresses both intra domain and inter domain distributions. Based on operational deployments, the current distribution model defined in [RFC4684] may cause some issue in specific scenarios.

This document refines the route distribution rules for inter domain NLRIs in order to address these specific scenarios.

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

Table of Contents

1. External NLRI propagation 2
  1.1. Peering type based pruning 3
  1.2. NLRI type based pruning 4
  1.3. Analysis of both approaches 4
2. Problem statement : disjoint peer AS 5
3. Proposal 6
4. Security considerations 7
5. Acknowledgements 7
6. IANA Considerations 7
7. Normative References 7
Authors' Addresses 8

1. External NLRI propagation

[RFC4684] Section 3.1 and 3.2 describes propagation of Route Target NLRI between ASes and inside an AS and distinguish two types of NLRIs :

- o Locally originated NLRI where origin-as field of the NLRI is equal to the local AS number.

- o External NLRI where origin-as field of the NLRI is different from the local AS number.

The global idea of inter AS propagation, is to propagate only VPN routes on shortest path towards the peer ASes using pruning of some branches of the distribution tree.

Based on current implementations of RFC4684, we can see two flavors of pruning for interAS that are both compatible with RFC4684 text.

- o Pruning based on peering type : pruning rule is applied when RT membership path are learned from eBGP peers only. No pruning is applied when path is iBGP.

- o Pruning based on NLRI type : pruning rule is applied to external RT membership NLRIs (source AS different from local AS). This pruning rule applies both to eBGP and iBGP.

1.1. Peering type based pruning

```
  AS 400                AS 500
                          |
         ASBR1 --- (mpebgp vpnv4+rtc)___
                |                        \
                |                         \
         ASBR2 --- (mpebgp vpnv4+rtc) -- PE1
                |                            \
                |                      (mpibgp vpnv4+rtc)
                                |                                    \
                |                                RR ------------ PE3
                |                               /
                |                      (mpibgp vpnv4+rtc)
                |                             /
         ASBR3 --- (mpebgp vpnv4+rtc) -- PE2
                |
                |
```

Figure 1

In the figure above, ASBR1,ASBR2 and ASBR3 are MPLS VPN nodes part of the AS 400. We consider that all these ASBRs are importing the same RT : 400:1, which is also exported by PE3. All ASBRs will generate the same RT membership NLRI 400:400:1/96 towards their PE. PE2 will send its path for this RT membership to RR. As PE1 has two ebgp paths for the same RT membership NLRI, it will apply pruning (as per peering type based pruning policy), if we consider that path from ASBR1 is the best path, RT distribution tree will only have a branch to ASBR1, and so ASBR2 will not receive any VPN route for RT 400:1 from PE1. PE1 will also send the RT membership NLRI to RR. RR will so have two paths for NLRI 400:400:1/96. As both path are iBGP, no pruning will be applied (as per peering type based pruning policy), and RR will create tree branches for 400:1 to both PE1 and PE2. As a result, VPN routes originated by PE3 with RT 400:1 will be sent by RR to PE1 and PE2. PE1 will propagate the routes only to ASBR1. PE2 will propagate the routes to ASBR3. AS 400 will have knowledge from PE3 routes only from ASBR1 and ASBR2.

1.2. NLRI type based pruning

We consider the same setup as in Figure 1. All ASBRs will generate the same RT membership NLRI 400:400:1/96 towards their PE. PE2 will send its path for this RT membership to RR. As PE1 has two ebgp paths for the same external RT membership NLRI, it will apply pruning (as per NLRI type based pruning policy, pruning is applied because NLRI is external), if we consider that path from ASBR1 is the best path, RT distribution tree will only have a branch to ASBR1, and so ASBR2 will not receive any VPN route for RT 400:1 from PE1. PE1 will also send the RT membership NLRI to RR. RR will so have two paths for NLRI 400:400:1/96. As the NLRI is external, pruning will be applied : if we consider that path from PE1 is the best one, a single branch of distribution tree will be added towards PE1. As a result, VPN routes originated by PE3 with RT 400:1 will be sent by RR to PE1 only. PE1 will propagate the routes only to ASBR1. AS 400 will have knowledge from PE3 routes only from ASBR1.

```
  AS 400                AS 500                      AS 400
           |                                  |
           |                                  |
           |                                  |
 cPE1 --------- sPE1 ------ RR ------- sPE2 ---------- cPE2
           |                                  |
           |                                  |
```

Figure 2

Figure 2 presents at typical case where an AS (AS400) uses another AS (AS500) as transit to build VPN services. If cPE1 and cPE2 shares a common VPN using RT 400:1, in case of NLRI type based pruning in AS500, RR in AS500 will perform pruning of VPN routes for NLRI 400:400:1/96. Considering that path from sPE1 is considered as best path, sPE2 will be pruned and cPE2 will never receive VPN routes from cPE1. This issue is discussed further in Section 2.

1.3. Analysis of both approaches

Both pruning approaches have pros and cons. Service Provider will need to be aware of this pros/cons while deploying inter AS RTC.

- o NLRI type based pruning helps in saving BGP paths in network nodes, inter AS distribution tree is only established on shortest path (at AS boundary and within the AS). In figure 1, PE2 does not receive VPN routes for RT 400:1 because these routes are already advertised through another path. This approach prevents hot potatoe routing and transit for disjoint ASes.

- o Peering type based pruning is based on the fact that the local AS does not know the precise location of the VPNs in the peer AS, so there is no reason for a route reflector to perform blind pruning that may lead to suboptimal routing. In figure 1, if we consider that ASBR3 is located in New York City, and ASBR1/2 are located in San Francisco. Considering that PE3 is located in Washington, performing NLRI type based pruning will prevent ASBR3 to receive PE3 routes, so routing from Washington to New York City will transit through San Francisco. We must note that in case of ASBR1 and ASBR2 being in two far cities, peering type based pruning will also suffer from suboptimal routing. The other point in favor of peering type pruning is faster convergence. In figure 1, when PE1 fails, backup routes are already available in AS400 through ASBR3.

As a summary, NLRI type based pruning helps in saving BGP paths in the transit networks, while peering type based pruning permits more optimal routing and faster convergence with the drawback of propagating additional routes. Peering type based pruning may also experience convergence or suboptimal routing case in case a single node is attached to multiple routers in the external AS.

2. Problem statement : disjoint peer AS

The previous section described how inter AS route distribution works and pros and cons of the existing approaches. Apart of these pros/cons, pruning in both solutions may lead to some problematic situation where the remote AS is disjoint, as already shown in Section 1.2.

```
+-------+
| DC1   | -- CE1 -- (mpebgp vpnv4+rtc) -- PE1
+-------+                                     \
                                       (mpibgp vpnv4+rtc)

    \
                                                 RR
                                                /
                                       (mpibgp vpnv4+rtc)
+-------+                                      /
| DC2   | -- CE2 -- (mpebgp vpnv4+rtc) -- PE2
+-------+
```

Figure 3

The figure above describes another typical service provider scenario where datacenters are connected through MPLS VPN interas option B with the Service Provider network. Route Target Constraint (RTC) is deployed on MPeBGP sessions as well as internally in the service provider network to ensure optimal distribution of VPN routes (required for scaling reason). In this scenario, both Datacenters

are using the same AS number, generally a private ASN (65000) like a typical PE-CE connection. As we expect DCs to communicate between each other, some features like "as-override" are deployed on PEs to overcome ASPATH loop issue.

In the Figure 3, CE1 and CE2 are advertising the RT 1:1 respectively to PE1 and PE2, the generated NLRI would be 65000:1:1/96. According to procedures defined in [RFC4684] Section 3.2, both PEs are using the standard BGP route selection and advertising rules. So both PEs are advertising their path for 65000:1:1/96 to the route-reflector. In case of NLRI type based pruning, route-reflector will establish the distribution tree only to PE1 (considering PE1 is the best path).

Due to this behavior, VPN routes from DC1 would never to send to DC2 because PE2 is not part of the flooding tree and as DC1 and DC2 are disjoint, even if they are using the same ASN, there is no communication possible between them.

The same issue may appear if two MPeBGP sites using the same ASN are connected on the same PE like in figure 4. In this situation both NLRI type based pruning and Peering type based pruning solutions are impacted.

```
+-------+
| DC1   |
+-------+
          \
                 (mpebgp vpnv4+rtc)
              \
                PE
              /
          (mpebgp vpnv4+rtc)
           /
+-------+
| DC2   |
+-------+
```

Figure 4

3. Proposal

This document proposes to introduce some new behavior in complement of [RFC4684] to manage the disjoint AS case.

In order to support our scenario, path pruning MAY be disabled by configuration for a given origin AS (different from the local AS). Implementations MAY also permit path pruning to be disabled for private AS numbers by default, but must make provision for it to be selectively enabled if such a feature is present.

This modification in establishing route distribution tree may create unnecessary flooding states in the situations where a real AS is multihomed to a service provider network (as displayed in Figure 3).

```
  ASN 65000                                                  ASN 64000
 +-----------+                                             +-------------+
 |   ASBR3   | -- (mpebgp vpnv4+rtc) -- ASBR1      PE1 ----  | CE1 --- DC1 |
 |     |     |                              \      /       +-------------+
 |     |     |                           (mpibgp vpnv4+rtc)
 |(vpnv4+rtc)|                                                      \  /
 |     |     |                                   RR
 |     |     |                                  /  \
 |     |     |                           (mpibgp vpnv4+rtc)          ASN 64000
 |     |     |                                /       \         +-------------+
 |   ASBR4   | -- (mpebgp vpnv4+rtc) -- ASBR2      PE2 ---- | CE2 --- DC2 |
 +-----------+                                             +-------------+
```

Figure 3

In the figure above, disabling pruning is required for AS64000 but it may be interesting to keep it enabled for AS65000. Implementations may require support for such granularity as proposed previously.

4. Security considerations

This document does not introduce any new security issue compared to [RFC4684].

5. Acknowledgements

6. IANA Considerations

There is no IANA consideration.

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, November 2006.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Jeff Haas
Juniper Networks

Email: jhaas@juniper.net

Keyur Patel
Cisco Systems

Email: keyupate@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 29, 2015

S. Previdi, Ed.
C. Filsfils
Cisco Systems, Inc.
S. Ray
Individual Contributor
K. Patel
Cisco Systems, Inc.
J. Dong
M. Chen
Huawei Technologies
April 27, 2015

Segment Routing Egress Peer Engineering BGP-LS Extensions
draft-previdi-idr-bgpls-segment-routing-epe-03

Abstract

Segment Routing (SR) leverages source routing. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with an SR header. A segment can represent any instruction, topological or service-based. SR allows to enforce a flow through any topological path and service chain while maintaining per-flow state only at the ingress node of the SR domain.

The Segment Routing architecture can be directly applied to the MPLS dataplane with no change on the forwarding plane. It requires minor extension to the existing link-state routing protocols.

This document outline a BGP-LS extension for exporting BGP egress point topology information (including its peers, interfaces and peering ASs) in a way that is exploitable in order to compute efficient Egress Point Engineering policies and strategies.

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 RFC 2119 [RFC2119].

Status of This Memo

working documents as Internet-Drafts. The list of current Internet-Drafts is at http://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 29, 2015.

Copyright Notice

Table of Contents

1. Introduction 3
2. Segment Routing Documents 3
3. BGP Peering Segments 3
4. Link NLRI for EPE Connectivity Description 4
  4.1. BGP Router ID and Member ASN 5
  4.2. EPE Node Descriptors 5
  4.3. Link Attributes 6
5. Peer Node and Peer Adjacency Segments 7
  5.1. Peer Node Segment 7
  5.2. Peer Adjacency Segment 9
  5.3. Peer Set Segment 10
6. Illustration 10
  6.1. Reference Diagram 10
    6.1.1. Peer Node Segment for Node D 12
    6.1.2. Peer Node Segment for Node H 12
    6.1.3. Peer Node Segment for Node E 12
    6.1.4. Peer Adj Segment for Node E, Link 1 13
    6.1.5. Peer Adj Segment for Node E, Link 2 13
7. BGP-LS EPE TLV/Sub-TLV Code Points Summary 14
8. IANA Considerations 14
9. Manageability Considerations 14

10. Security Considerations 14
11. Acknowledgements 15
12. References 15
12.1. Normative References 15
12.2. Informative References 15
Authors' Addresses 16

1. Introduction

Segment Routing (SR) leverages source routing. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with an SR header. A segment can represent any instruction, topological or service-based. SR allows to enforce a flow through any topological path and service chain while maintaining per-flow state only at the ingress node of the SR domain.

The Segment Routing architecture can be directly applied to the MPLS dataplane with no change on the forwarding plane. It requires minor extension to the existing link-state routing protocols.

This document outline a BGP-LS extension for exporting BGP egress point topology information (including its peers, interfaces and peering ASs) in a way that is exploitable in order to compute efficient Egress Point Engineering policies and strategies.

This document defines new types of segments: a Peer Node segment describing the BGP session between two nodes; a Peer Adjacency Segment describing the link (one or more) that is used by the BGP session; the Peer Set Segment describing an arbitrary set of sessions or links between the local BGP node and its peers.

2. Segment Routing Documents

The main reference for this document is the SR architecture defined in [I-D.ietf-spring-segment-routing].

The Segment Routing Egress Peer Engineering architecture is described in [I-D.filsfils-spring-segment-routing-central-epe].

3. BGP Peering Segments

As defined in [draft-filsfils-spring-segment-routing-epe], an EPE enabled Egress PE node MAY advertise segments corresponding to its attached peers. These segments are called BGP peering segments or BGP Peering SIDs. They enable the expression of source-routed inter-domain paths.

An ingress border router of an AS may compose a list of segments to steer a flow along a selected path within the AS, towards a selected egress border router C of the AS and through a specific peer. At minimum, a BGP Peering Engineering policy applied at an ingress PE involves two segments: the Node SID of the chosen egress PE and then the BGP Peering Segment for the chosen egress PE peer or peering interface.

This document defines the BGP EPE Peering Segments: Peer Node, Peer Adjacency and Peer Set.

Each BGP session MUST be described by a Peer Node Segment. The description of the BGP session MAY be augmented by additional Adjacency Segments. Finally, each Peer Node Segment and Peer Adjacency Segment MAY be part of the same group/set so to be able to group EPE resources under a common Peer-Set Segment Identifier (SID).

Therefore, when the extensions defined in this document are applied to the use case defined in [I-D.filsfils-spring-segment-routing-central-epe]:

- o One Peer Node Segment MUST be present.

- o One or more Peer Adjacency Segments MAY be present.

- o Each of the Peer Node and Peer Adjacency Segment MAY use the same Peer-Set.

4. Link NLRI for EPE Connectivity Description

This section describes the NLRI used for describing the connectivity of the BGP Egress router. The connectivity is based on links and remote peers/ASs and therefore the existing Link-Type NLRI (defined in [I-D.ietf-idr-ls-distribution]) is used. A new Protocol ID is used (codepoint to be assigned by IANA, suggested value 7).

The use of a new Protocol-ID allows separation and differentiation between the NLRIs carrying BGP-EPE descriptors from the NLRIs carrying IGP link-state information as defined in[I-D.ietf-idr-ls-distribution]. The Link NLRI Type uses descriptors and attributes already defined in [I-D.ietf-idr-ls-distribution] in addition to new TLVs defined in the following sections of this document.

The format of the Link NLRI Type is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|  Protocol-ID  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Identifier                          |
|                            (64 bits)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//      Local Node Descriptors                                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//      Remote Node Descriptors                                 //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//      Link Descriptors                                        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Node Descriptors and Link Descriptors are defined in [I-D.ietf-idr-ls-distribution].

4.1. BGP Router ID and Member ASN

Two new Node Descriptors Sub-TLVs are defined in this document:

- o BGP Router Identifier (BGP Router-ID):

  Type: TBA (suggested value 516).

  Length: 4 octets

  Value: 4 octet unsigned integer representing the BGP Identifier as defined in [RFC4271] and [RFC6286].

- o Confederation Member ASN (Member-ASN)

  Type: TBA (suggested value 517).

  Length: 4 octets

  Value: 4 octet unsigned integer representing the Member ASN inside the Confederation.[RFC5065].

4.2. EPE Node Descriptors

The following Node Descriptors Sub-TLVs MUST appear in the Link NLRI as Local Node Descriptors:

- o BGP Router ID, which contains the BGP Identifier of the local BGP EPE node.

- o Autonomous System Number, which contains the local ASN or local confederation identifier (ASN) if confederations are used.

- o BGP-LS Identifier.

The following Node Descriptors Sub-TLVs MAY appear in the Link NLRI as Local Node Descriptors:

- o Member-ASN, which contains the ASN of the confederation member (when BGP confederations are used).

- o Node Descriptors as defined in [I-D.ietf-idr-ls-distribution].

The following Node Descriptors Sub-TLVs MUST appear in the Link NLRI as Remote Node Descriptors:

- o BGP Router ID, which contains the BGP Identifier of the peer node.

- o Autonomous System Number, which contains the peer ASN or the peer confederation identifier (ASN), if confederations are used.

The following Node Descriptors Sub-TLVs MAY appear in the Link NLRI as Remote Node Descriptors:

- o Member-ASN, which contains the ASN of the confederation member (when BGP confederations are used).

- o Node Descriptors as defined in defined in [I-D.ietf-idr-ls-distribution].

4.3. Link Attributes

The following BGP-LS Link attributes TLVs are used with the Link NLRI:

TLV Code Point	Description	Length
1099	Adjacency-Segment Identifier (Adj-SID)	variable
TBA	Peer-Segment Identifier (Peer-SID)	variable
TBA	Peer-Set-SID	variable

Adj-SID is defined in [I-D.gredler-idr-bgp-ls-segment-routing-extension] and the same format is used for the Peer-SID and Peer-Set-SID TLVs.

Peer-SID and Peer-Set SID are two new sub-TLVs with the same format as the Adj-SID and whose codepoints are to be assigned by IANA:

Peer-SID: SID representing the peer of the BGP session. The format is the same as defined for the Adj-SID in [I-D.gredler-idr-bgp-ls-segment-routing-extension]. Suggested codepoint value: 1036

Peer-Set-SID: the SID representing the group the peer is part of. The format is the same as defined for the Adj-SID in [I-D.gredler-idr-bgp-ls-segment-routing-extension]. Suggested codepoint value: 1037

The value of the Adj-SID, Peer-SID and Peer-Set-SID Sub-TLVs SHOULD be persistent across router restart.

The Peer-SID MUST be present when BGP-LS is used for the use case described in [I-D.filsfils-spring-segment-routing-central-epe] and MAY be omitted for other use cases.

The Adj-SID and Peer-Set-SID SubTLVs MAY be present when BGP-LS is used for the use case described in [I-D.filsfils-spring-segment-routing-central-epe] and MAY be omitted for other use cases.

In addition, BGP-LS Nodes and Link Attributes, as defined in [I-D.ietf-idr-ls-distribution]MAY be inserted in order to advertise the characteristics of the link.

5. Peer Node and Peer Adjacency Segments

In this section the following Peer Segments are defined:

Peer Node Segment (Peer Node SID)

Peer Adjacency Segment (Peer Adj SID)

Peer Set Segment (Peer Set SID)

5.1. Peer Node Segment

The Peer Node Segment describes the BGP session peer (neighbor). It MUST be present when describing an EPE topology as defined in [I-D.filsfils-spring-segment-routing-central-epe]. The Peer Node

Segment is encoded within the BGP-LS Link NLRI specified in Section 4.

The Peer Node Segment is a local segment. At the BGP node advertising it, its semantic is:

- o SR header operation: NEXT (as defined in [I-D.ietf-spring-segment-routing]).

- o Next-Hop: the connected peering node to which the segment is related.

The Peer Node Segment is advertised with a Link NLRI, where:

- o Local Node Descriptors contains

  Local BGP Router ID of the EPE enabled egress PE.
  Local ASN.
  BGP-LS Identifier.

- o Remote Node Descriptors contains

  Peer BGP Router ID (i.e.: the peer BGP ID used in the BGP session).
  Peer ASN.

- o Link Descriptors Sub-TLVs, as defined in [I-D.ietf-idr-ls-distribution], contain the addresses used by the BGP session:

  * IPv4 Interface Address (Sub-TLV 259) contains the BGP session IPv4 local address.

  * IPv4 Neighbor Address (Sub-TLV 260) contains the BGP session IPv4 peer address.

  * IPv6 Interface Address (Sub-TLV 261) contains the BGP session IPv6 local address.

  * IPv6 Neighbor Address (Sub-TLV 262) contains the BGP session IPv6 peer address.

- o Link Attribute contains the Peer-SID TLV as defined in Section 4.3.

- o In addition, BGP-LS Link Attributes, as defined in [I-D.ietf-idr-ls-distribution], MAY be inserted in order to advertise the characteristics of the link.

5.2. Peer Adjacency Segment

The Peer Adjacency Segment is a local segment. At the BGP node advertising it, its semantic is:

- o SR header operation: NEXT (as defined in [I-D.ietf-spring-segment-routing]).

- o Next-Hop: the interface peer address.

The Peer Adjacency Segment is advertised with a Link NLRI, where:

- o Local Node Descriptors contains

  Local BGP Router ID of the EPE enabled egress PE.
  Local ASN.
  BGP-LS Identifier.

- o Remote Node Descriptors contains

  Peer BGP Router ID (i.e.: the peer BGP ID used in the BGP session).
  Peer ASN.

- o Link Descriptors Sub-TLVs, as defined in [I-D.ietf-idr-ls-distribution], contain the addresses used by the BGP session:

  * IPv4 Interface Address (Sub-TLV 259) contains the IPv4 address of the local interface used by the BGP session.

  * IPv4 Neighbor Address (Sub-TLV 260) contains the IPv4 address of the peer interface used by the BGP session.

  * IPv6 Interface Address (Sub-TLV 261) contains the IPv6 address of the local interface used by the BGP session.

  * IPv6 Neighbor Address (Sub-TLV 262) contains the IPv6 address of the peer interface used by the BGP session.

- o Link attribute used with the Peer Adjacency SID contains the Adj-SID TLV as defined in Section 4.3.

In addition, BGP-LS Link Attributes, as defined in [I-D.ietf-idr-ls-distribution], MAY be inserted in order to advertise the characteristics of the link.

5.3. Peer Set Segment

The Peer Set Segment is a local segment. At the BGP node advertising it, its semantic is:

- o SR header operation: NEXT (as defined in [I-D.ietf-spring-segment-routing]).

- o Next-Hop: load balance across any connected interface to any peer in the related set.

The Peer Set Segment is advertised within a Link NLRI (describing a Peer Node Segment or a Peer Adjacency segment) as a BGP-LS attribute.

The Peer Set Attribute contains the Peer-Set-SID TLV, defined in Section 4.3 identifying the set of which the Peer Node Segment or Peer Adjacency Segment is a member.

6. Illustration

6.1. Reference Diagram

The following reference diagram is used throughout this document. The solution is described for IPv4 with MPLS-based segments.

```
                  +------+
                  |      |
             +---D      F
+---------+ /    | AS 2 |\  +------+
|    X    |/     +------+ \ |   Z  |---L/8
A         C---+            \|      |
|         |\\  \  +------+ /| AS 4 |---M/8
|   AS1   | \\  +-H      |/ +------+
|         |  \\   |      G
+----P----+   +===E AS 3 |
     |              +--Q---+
     |                 |
     +----------------+
```

Figure 1: Reference Diagram

IPv4 addressing:

- o C's interface to D: 1.0.1.1/24, D's interface: 1.0.1.2/24

- o C's interface to H: 1.0.2.1/24, H's interface: 1.0.2.2/24

- o C's upper interface to E: 1.0.3.1/24, E's interface: 1.0.3.2/24

- o C's lower interface to E: 1.0.4.1/24, E's interface: 1.0.4.2/24
- o Loopback of E used for eBGP multi-hop peering to C: 1.0.5.2/32
- o C's loopback is 3.3.3.3/32 with SID 64

BGP Router-IDs are C, D, H and E.

- o C's BGP Router-ID: 3.3.3.3
- o D's BGP Router-ID: 4.4.4.4
- o E's BGP Router-ID: 5.5.5.5
- o H's BGP Router-ID: 6.6.6.6

C's BGP peering:

- o Single-hop eBGP peering with neighbor 1.0.1.2 (D)
- o Single-hop eBGP peering with neighbor 1.0.2.2 (H)
- o Multi-hop eBGP peering with E on ip address 1.0.5.2 (E)

C's resolution of the multi-hop eBGP session to E:

- o Static route 1.0.5.2/32 via 1.0.3.2
- o Static route 1.0.5.2/32 via 1.0.4.2

Node C configuration is such that:

- o A Peer Node segment is allocated to each peer (D, H and E).
- o An Adjacency segment is defined for each recursing interface to a multi-hop peer (CE upper and lower interfaces).
- o A Peer Set segment is defined to include all peers in AS3 (peers H and E).

Local BGP-LS Identifier in router C is set to 10000.

The Link NLRI Type is used in order to encode C's connectivity. the Link NLRI uses the new Protocol-ID value (to be assigned by IANA).

6.1.1. Peer Node Segment for Node D

Descriptors:

- o Local Node Descriptors (BGP Router-ID, local ASN, BGP-LS Identifier): 3.3.3.3 , AS1, 10000
- o Remote Node Descriptors (BGP Router-ID, peer ASN): 4.4.4.4, AS2
- o Link Descriptors (BGP session IPv4 local address, BGP session IPv4 neighbor address): 1.0.1.1, 1.0.1.2

Attributes:

- o Peer-SID: 1012
- o Link Attributes: see section 3.3.2 of [I-D.ietf-idr-ls-distribution]

6.1.2. Peer Node Segment for Node H

Descriptors:

- o Local Node Descriptors (BGP Router-ID, ASN, BGPL Identifier): 3.3.3.3 , AS1, 10000
- o Remote Node Descriptors (BGP Router-ID ASN): 6.6.6.6, AS3
- o Link Descriptors (BGP session IPv4 local address, BGP session IPv4 peer address): 1.0.2.1, 1.0.2.2

Attributes:

- o Peer-SID: 1022
- o Peer-Set-SID: 1060
- o Link Attributes: see section 3.3.2 of [I-D.ietf-idr-ls-distribution]

6.1.3. Peer Node Segment for Node E

Descriptors:

- o Local Node Descriptors (BGP Router-ID, ASN, BGP-LS Identifier): 3.3.3.3 , AS1, 10000
- o Remote Node Descriptors (BGP Router-ID, ASN): 5.5.5.5, AS3

- o Link Descriptors (BGP session IPv4 local address, BGP session IPv4 peer address): 3.3.3.3, 1.0.5.2

Attributes:

- o Peer-SID: 1052
- o Peer-Set-SID: 1060

6.1.4. Peer Adj Segment for Node E, Link 1

Descriptors:

- o Local Node Descriptors (BGP Router-ID, ASN, BGP-LS Identifier): 3.3.3.3 , AS1, 10000
- o Remote Node Descriptors (BGP Router-ID, ASN): 5.5.5.5, AS3
- o Link Descriptors (IPv4 local interface address, IPv4 peer interface address): 1.0.3.1 , 1.0.3.2

Attributes:

- o Adj-SID: 1032
- o LinkAttributes: see section 3.3.2 of [I-D.ietf-idr-ls-distribution]

6.1.5. Peer Adj Segment for Node E, Link 2

Descriptors:

- o Local Node Descriptors (BGP Router-ID, ASN, BGP-LS Identifier): 3.3.3.3 , AS1, 10000
- o Remote Node Descriptors (BGP Router-ID, ASN): 5.5.5.5, AS3
- o Link Descriptors (IPv4 local interface address, IPv4 peer interface address): 1.0.4.1 , 1.0.4.2

Attributes:

- o Adj-SID: 1042
- o LinkAttributes: see section 3.3.2 of [I-D.ietf-idr-ls-distribution]

7. BGP-LS EPE TLV/Sub-TLV Code Points Summary

The following table contains the TLVs/Sub-TLVs defined in this document.

Suggested Codepoint	Description	Defined in:
7	Protocol-ID	Section 4
516	BGP Router ID	Section 4.1
517	BGP Confederation Member	Section 4.1
1036	Peer-SID	Section 4.3
1037	Peer-Set-SID	Section 4.3

Table 1: Summary Table of BGP-LS EPE Codepoints

8. IANA Considerations

This document defines:

Two new Node Descriptors Sub-TLVs: BGP-Router-ID and BGP Confederation Member.

A new Protocol-ID for EPE: BGP-EPE.

Two new BGP-LS Attribute Sub-TLVs: the Peer-SID and the Peer-Set-SID.

The codepoints are to be assigned by IANA.

9. Manageability Considerations

TBD

10. Security Considerations

[I-D.ietf-idr-ls-distribution] defines BGP-LS NLRIs to which the extensions defined in this document apply.

The Security Section of [I-D.ietf-idr-ls-distribution] also applies to the:

new Node Descriptors Sub-TLVs: BGP-Router ID and BGP Confederation Member;

Peer-SID and Peer-Set-SID attributes

defined in this document.

11. Acknowledgements

The authors would like to thank Acee Lindem, Jakob Heitz, Howard Yang and Hannes Gredler for their feedback and comments.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, August 2007.

[RFC6286] Chen, E. and J. Yuan, "Autonomous-System-Wide Unique BGP Identifier for BGP-4", RFC 6286, June 2011.

12.2. Informative References

[I-D.filsfils-spring-segment-routing-central-epe]
Filsfils, C., Previdi, S., Patel, K., Aries, E., shaw@fb.com, s., Ginsburg, D., and D. Afanasiev, "Segment Routing Centralized Egress Peer Engineering", draft-filsfils-spring-segment-routing-central-epe-03 (work in progress), January 2015.

[I-D.gredler-idr-bgp-ls-segment-routing-extension]
Gredler, H., Ray, S., Previdi, S., Filsfils, C., Chen, M., and J. Tantsura, "BGP Link-State extensions for Segment Routing", draft-gredler-idr-bgp-ls-segment-routing-extension-02 (work in progress), October 2014.

[I-D.ietf-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-10 (work in progress), January 2015.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Shakir, R., Tantsura, J., and E. Crabbe, "Segment Routing Architecture", draft-ietf-spring-segment-routing-01 (work in progress), February 2015.

Authors' Addresses

Stefano Previdi (editor)
Cisco Systems, Inc.
Via Del Serafico, 200
Rome 00142
Italy

Email: sprevidi@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Saikat Ray
Individual Contributor

Email: raysaikat@gmail.com

Keyur Patel
Cisco Systems, Inc.
170, West Tasman Drive
San Jose, CA 95134
US

Email: keyupate@cisco.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Mach (Guoyi) Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: mach.chen@huawei.com

Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: May 18, 2017

P. Sarkar, Ed.
Individual Contributor
H. Gredler
RtBrick, Inc.
S. Litkowski
Orange

November 14, 2016

Advertising Node Admin Tags in BGP Link-State Advertisements
draft-psarkar-idr-bgp-ls-node-admin-tag-extension-05

Abstract

This document describes the protocol extensions to collect node administrative tags adevertised in IGP Link State advertisements and disseminate the same in BGP Link-State advertisement protocol, to facilitate inter-AS TE applications that may need the same node administrative tags to associate a subset of network devices spanning across more than one AS with a specific functionality.

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 RFC 2119 [RFC2119].

Status of This Memo

Table of Contents

1. Introduction 2
2. Per-Node Administrative Tag 4
3. BGP-LS Extensions for Per-Node Administrative Tags 5
3.1. Node Admin Tag TLV 5
4. Elements of Procedure 7
5. Applications 8
6. IANA Considerations 8
7. Manageability Considerations 9
7.1. Operational Considerations 9
7.1.1. Operations 9
8. TLV/Sub-TLV Code Points Summary 9
9. Security Considerations 9
10. Acknowledgements 9
11. References 9
11.1. Normative References 9
11.2. Informative References 10
11.3. URIs 10
Authors' Addresses 10

1. Introduction

Advertising Node Administrative Tags in Link State protocols like IS-IS [RFC7917] and OSPF [RFC7777] allows adding an optional operational capability, that allows tagging and grouping of the nodes in a IGP domain. This, among other applications, allows simple management and easy control over route and path selection, based on local configured policies. However node administrative tags advertised in IGP advertisements let network operators associate nodes within a single AS (if not a single area). This limits the use of such node administrative tags and applications that need to associate a subset

of network devices spanning across multiple AS with a specific functionality cannot use them.

To address the need for applications that require visibility into LSDB across IGP areas, or even across ASes, the BGP-LS address-family/sub-address-family have been defined that allows BGP to carry LSDB information. The BGP Network Layer Reachability Information (NLRI) encoding format for BGP-LS and a new BGP Path Attribute called BGP-LS attribute are defined in [RFC7752]. The identifying key of each LSDB object, namely a node, a link or a prefix, is encoded in the NLRI and the properties of the object are encoded in the BGP-LS attribute. Figure 1 describes a typical deployment scenario. In each IGP area, one or more nodes are configured with BGP-LS. These BGP speakers form an IBGP mesh by connecting to one or more route-reflectors. This way, all BGP speakers - specifically the route-reflectors - obtain LSDB information from all IGP areas (and from other ASes from EBGP peers). An external component connects to the route-reflector to obtain this information (perhaps moderated by a policy regarding what information is sent to the external component, and what information isn't).

```
                          +------------+
                          |  Consumer  |
                          +------------+
                                ^
                                |
                                v
                     +--------------------+
                     |    BGP Speaker     |          +-----------+
                     | (Route-Reflector)  |          | Consumer  |
                     +--------------------+          +-----------+
                           ^   ^   ^                       ^
                           |   |   |                       |
            +---------------+   |   +-------------------+   |
            |                   |                       |   |
            v                   v                       v   v
       +-----------+       +-----------+             +-----------+
       |    BGP    |       |    BGP    |             |    BGP    |
       |  Speaker  |       |  Speaker  |    . . .    |  Speaker  |
       +-----------+       +-----------+             +-----------+
             ^                   ^                         ^
             |                   |                         |
            IGP                 IGP                       IGP
```

Figure 1: Link State info collection

For the purpose of advertising node administrative tags within BGP Link-State advertisements, a new Node Attribute TLV to be carried in

the corresponding BGP-LS Node NLRI is proposed. For more details on the Node Attribute TLVs please refer to section 3.3.1 in [RFC7752]

2. Per-Node Administrative Tag

An administrative Tag is a 32-bit integer value that can be used to identify a group of nodes in the entire routing domain. The new sub-TLV specifies one or more administrative tag values. A BGP Link-State speaker that also participates in the IGP link state advertisements exchange may learn one or more node administrative tags advertised by another router in the same IGP domain. Such BGP-LS speaker shall encode the same set of node administrative tags in the corresponding Node Attribute TLV representing the network device that originated the node administrative tags.

The node administrative tags advertised in IGP link state advertisements will have either per-area(or levels in IS-IS)scope or 'global' scope. Operator may choose to a set of node administrative tags across areas (or levels in IS-IS) and another advertise set of node administrative tags within the specific area (or level). But evidently two areas within the same AS or two different may use the same node administrative tag for different purposes. In such case applications will need to distinguish between the per-area(or level) scoped administrative tags originated from a specific node against those originated from the same node with 'global' scope.

A BGP-LS router in a given AS while copying the node administrative tags learnt from IGP link-state advertisements, MUST also copy the scope associated with the node administrative tags. Refer to Section 3.1 for how to encode the associated scope of a node administrative tags as well.

To be able to distinguish between the significance of a per-area(or level) administrative tag learnt in one area, from that advertised in another area, or another AS, any applications receiving such a BGP-LS advertisements MUST consider the scope associated with each node administrative tag with 'per-area (or per-level) along with the area(or level in IS-IS) associated with corresponding IGP link state advertisement and the AS number associated with the originating node. The area(or level) associated with corresponding IGP link state advertisement and the AS number associated with the originating node can be derived from appropriate node attributes (already defined in BGP-LS [RFC7752]) attached with the corresponding Node NLRI.

3. BGP-LS Extensions for Per-Node Administrative Tags

The BGP-LS NLRI can be a node NLRI, a link NLRI or a prefix NLRI. The corresponding BGP-LS attribute is a node attribute, a link attribute or a prefix attribute. BGP-LS [RFC7752] defines the TLVs that map link-state information to BGP-LS NLRI and BGP-LS attribute. This document adds an new Node Attribute TLV called 'Node Admin Tag TLV' to encode node administrative tags information.

[RFC7917] defines the 'Node Admin Tag' sub-TLV in the Router Capability TLV (type 242) in IS-IS Link State PDUs to encode node administrative tags. Similarly [RFC7917] defines the 'Node Administrative Tag' TLV in OSPF Router Information LSAs to encode node administrative tags in OSPF Link State update packets. The node administrative tags TLVs learnt from the IGP link state advertisements of a specific node will all be inserted in a new Node Admin Tag TLV and added to the corresponding Node are mapped to the corresponding BGP-LS Node NLRI. Node administrative tags from IGP advertisements are mapped to the corresponding Node Admin Tag TLV in the following way.

TLV Code Point	Description	Length	IS-IS TLV /sub-TLV	OSPF LSA/TLV
TBD	Node Admin Tag TLV	Variable	242/TBD [1]	RI-LSA/TBD [2]

Table 1: Node Admin Tag TLV Mapping from IGP

3.1. Node Admin Tag TLV

The new Node Administrative Tag TLV, like other BGP-LS Node Attribute TLVs, is formatted as Type/Length/Value (TLV)triplets. Figure 2 below shows the format of the new TLV.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Flags              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Administrative Tag #1                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Administrative Tag #2                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                                                             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Administrative Tag #N                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type : A 2-octet field specifiying code-point of the new TLV type. Code-point: TBA (suggested 1040)

Length: A 2-octet field that indicates the length of the value portion in octets and will be a multiple of 4 octets dependent on the number of tags advertised.

Value: A 2-octet 'Flags' field, followed by a sequence of multiple 4 octets defining the administrative tags.

Flags: A 2-octet field that carries flags associated with all the administrative flags encoded in this TLV. Following is the format of this field.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|            Reserved         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following bit flags are defined:

L bit : If the L bit is set (1), it signifies that all administrative flags encoded in this TLV has per-area(or level in IS-IS) scope, and should not be mixed with ones with same value but with 'global' scope (L bit reset to 0).

Figure 2: BGP Link-State Node Administrative Tag TLV

This new type of 'Node Admin Tag' TLVs can ONLY be added to the Node Attribute associated with the Node NLRI that originates the corresponding node administrative tags in IGP domain.

All the node administrative tags with 'per-area' (or per-level) scope, originated by a single node in IGP domain SHALL be re-originated in a single 'Node Admin Tag' TLV and inserted in the Node NLRI generated for the same node. Similarly, all the node administrative tags with 'global' scope originated by the same node in IGP domain SHALL be re-originated in another 'Node Admin Tag' TLV and inserted in the same Node NLRI generated for the originating node. Multiple instances of a TLV may be generated by the BGP-lS router for a given node in the IGP domain. This MAY happen if the original node's link state advertisement carries more than 16383 node administrative groups and a single TLV does not provide sufficient space. As such multiple occurence of the 'Node Admin Tag' TLVs under a single BGP LS NLRI is cumulative.

While copying node administrative tags from IGP link-state advertisements to corresponding BGP-LS advertisements, the said BGP-LS speaker MAY run all the node administrative flags through a locally configured policy that selects which ones should be exported and which ones not. And then the node administrative tag is copied to the BGP-LS advertisement if it is permitted to do so by the said policy.

4. Elements of Procedure

Meaning of the Node administrative tags is generally opaque to BGP Link-State protocol. Router advertising the node administrative tag (or tags) may be configured to do so without knowing (or even explicitly supporting) functionality implied by the tag.

Interpretation of tag values is specific to the administrative domain of a particular network operator. The meaning of a node administrative tag is defined by the network local policy and is controlled via the configuration. However multiple administrative domain owners may agree on a common meaning implied by a administrative tag for mutual benefit.

The semantics of the tag order has no meaning. There is no implied meaning to the ordering of the tags that indicates a certain operation or set of operations that need to be performed based on the ordering.

Each tag SHOULD be treated as an independent identifier that MAY be used in policy to perform a policy action. Node administrative tags carried by the Node Admin Tag TLV SHOULD be used to indicate a

independent characteristics of the node in IGP domain that originated it. The TLV SHOULD be considered as an unordered list. Whilst policies may be implemented based on the presence of multiple tags (e.g., if tag A AND tag B are present), they MUST NOT be reliant upon the order of the tags (i.e., all policies should be considered commutative operations, such that tag A preceding or following tag B does not change their outcome).

For more details on guidance on usage of node administrative tags please refer to section 4 [3] in [RFC7917].

5. Applications

[RFC7917] and [RFC7777] present some applications of node administrative tags.

The Policy-based Explicit routing use case can be extended to inter-area or inter-AS scenarios where an end to end path needs to avoid or include nodes that have particular properties. Following are some examples.

1. Geopolitical routing : preventing traffic from country A to country B to cross country C. In this case, we may use node administrative tags to encode geographical information (country). Path computation will be required to take into account node administrative tag to permit avoidance of nodes belonging to country C.

2. Legacy node avoidance : in some specific cases, it is interesting for service-provider to force some traffic to avoid legacy nodes in the network. For example, legacy nodes may not be carrier class (no high availability), and service provider wants to ensure that critical traffic only uses nodes that are providing high availability.

In case of inter-AS Traffic-Engineering applications, different ASes SHOULD share their administrative tag policies. They MAY also need to agree upon some common tagging policy for specific applications.

For more details on some possible applications with node administrative tags please refer to section 3 [4] in [RFC7777].

6. IANA Considerations

This document requests assigning code-points from the registry for BGP-LS attribute TLVs based on table Table 2.

7. Manageability Considerations

This section is structured as recommended in [RFC5706].

7.1. Operational Considerations

7.1.1. Operations

Existing BGP and BGP-LS operational procedures apply. No new operation procedures are defined in this document.

8. TLV/Sub-TLV Code Points Summary

This section contains the global table of all TLVs/Sub-TLVs defined in this document.

TLV Code Point	Description	Length
1040	Node Admin Tag	variable

Table 2: Summary Table of TLV/Sub-TLV Codepoints

9. Security Considerations

Procedures and protocol extensions defined in this document do not affect the BGP security model. See the 'Security Considerations' section of [RFC4271] for a discussion of BGP security. Also refer to [RFC4272] and [RFC6952] for analysis of security issues for BGP.

10. Acknowledgements

TBA.

11. References

11.1. Normative References

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

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <http://www.rfc-editor.org/info/rfc4271>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <http://www.rfc-editor.org/info/rfc7752>.

11.2. Informative References

[RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis", RFC 4272, DOI 10.17487/RFC4272, January 2006, <http://www.rfc-editor.org/info/rfc4272>.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, DOI 10.17487/RFC5706, November 2009, <http://www.rfc-editor.org/info/rfc5706>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <http://www.rfc-editor.org/info/rfc6952>.

[RFC7777] Hegde, S., Shakir, R., Smirnov, A., Li, Z., and B. Decraene, "Advertising Node Administrative Tags in OSPF", RFC 7777, DOI 10.17487/RFC7777, March 2016, <http://www.rfc-editor.org/info/rfc7777>.

[RFC7917] Sarkar, P., Ed., Gredler, H., Hegde, S., Litkowski, S., and B. Decraene, "Advertising Node Administrative Tags in IS-IS", RFC 7917, DOI 10.17487/RFC7917, July 2016, <http://www.rfc-editor.org/info/rfc7917>.

11.3. URIs

[1] http://tools.ietf.org/html/rfc7917#section-3.1

[2] http://tools.ietf.org/html/rfc7777#section-2.1

[3] http://tools.ietf.org/html/rfc7917#section-4

[4] http://tools.ietf.org/html/rfc7777#section-3

Authors' Addresses

Pushpasis Sarkar (editor)
Individual Contributor

Email: pushpasis.ietf@gmail.com

Hannes Gredler
RtBrick, Inc.

Email: hannes@rtbrick.com

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2015

R. Raszuk, Ed.
Mirantis Inc.
J. Haas
Juniper Networks

March 7, 2015

Registered Wide BGP Community Values
draft-raszuk-registered-wide-bgp-communities-00

Abstract

Communicating various routing policies via route tagging plays an important role in external BGP peering relations. The most common tool used today to attach various information about routes is realized with the use of BGP communities. Such information is important for the peering AS to perform some mutually agreed actions without the need to maintain a separate offline database for each pair of prefix and an associated with it requested set of action entries.

This document proposes to establish a new IANA maintained registry of most commonly used Wide BGP Communities by network operators. Such public registry will allow for easy refernece and clear interpretation of the actions associated with received community values.

Status of This Memo

Table of Contents

1. Introduction 2
2. Globally significant pre-defined values 3
2.1. Well Known Standard BGP Communities 3
2.2. Registered pre-defined Wide BGP Communities 3
2.2.1. General Registered Wide BGP Community Values 4
2.2.2. Advertisement control Registered Wide BGP Communities 7
2.2.3. AS source marking Registered Wide BGP Communities . . 8
2.2.4. Return path influencing Registered Wide BGP Communities 10
2.2.5. AS_PATH modifying Registered Wide BGP Communities . . 10
2.2.6. Local Preference Registered Community 11
2.2.7. AS_PATH TTL Registered Community 12
2.2.8. GEO-LOCATION Registered Community 12
3. Example 13
4. Security considerations 14
5. IANA Considerations 14
6. Contributors 15
7. Acknowledgments 16
8. References 16
8.1. Normative References 16
8.2. Informative References 17
Authors' Addresses 17

1. Introduction

RFC 1997 [RFC1997] defines a BGP Community Attribute to be used as a tool to contain in BGP update message various additional information about routes which may help to automate peering administration. As defined in RFC 1997 [RFC1997] BGP Communities attribute consists of one or more sets of four octet values, where each one of them specifies a different community. Except two reserved ranges the

encoding of community values mandates that first two octets are to contain the Autonomous System number followed by next two octets containing locally defined value.

This document lists the most commonly used today BGP communities as well as provides a new registry for future definitions.

2. Globally significant pre-defined values

2.1. Well Known Standard BGP Communities

According to RFC 1997 as well as to IANA's Well-Known BGP Communities registry today the following BGP communities are defined to have global significance:

0xFFFF0000	planned-shut	[draft-francois-bgp-gshut]
0xFFFFFF01	NO_EXPORT	[RFC1997]
0xFFFFFF02	NO_ADVERTISE	[RFC1997]
0xFFFFFF03	NO_EXPORT_SUBCONFED	[RFC1997]
0xFFFFFF04	NOPEER	[RFC3765]

This document recommends for simplicity as well as for avoidance of backward compatibility issues the continued use of BGP Standard Community Attribute type 8 as defined in RFC 1997 to distribute non Autonomous System specific Well-Known BGP Communities.

For the same reason the described registry does not intend to obsolete BGP Extended Community Attribute and any already defined and deployed extended communities. The new registry is to be used primarly for new community definitions in particular those which require to carry various new parameters or which should be propagated with a controled scope and radius.

2.2. Registered pre-defined Wide BGP Communities

It has been requested numerous times to have a globally unified way to express some particular Autonomous System based routing policies. When defining a new way to encode bgp communities we have an opportunity to define set of new registered routing policies and route markings which could be passed within and between Autonomous Systems resulting in their common interpretation.

This document will request IANA to define and maintain a new registry for pre-defined Wide BGP Community values. The allocation policy is on a first come first served basis.

It is recommended that an implementation supports by an explicit enabling defined below Registered Wide BGP Communities. Depending on the BGP implementation support it is recommended that an implementation would support Registered Wide BGP Communities without breaking static or dynamic peer/update groups. However it needs to be pointed out that support of all Registered Wide BGP Communities is not mandatory. It will be perfectly valid for any BGP implementation to support only subset of Wide BGP Communities.

It is strongly advised that each Autonomous System does an inbound verification of received Wide BGP Communities from all of its EBGP peers before accepting them and propagating within their own domain.

The document does not mandate nor enforces that given registered type value of Wide BGP Community would be of transitive or non-transitive type. It is for the operator to determine the propagation AS radius required for such community when appending it to routing information. However the document will provide a transitivity radius recommendation to defined communities.

The following Wide BGP Communities have global significance and their execution should be uniformly implemented by any BGP speaker supporting given set of Wide BGP Communities.

The defined below value of the community should be interpreted as registered value only if "R" - registered bit is set in the community Type 1 container as described in [draft-raszuk-wide-bgp-communities] Otherwise the value is local and it’s actions is locally defined by the operator.

2.2.1. General Registered Wide BGP Community Values

The below set of communities will be defined to be carried in Wide BGP Community Type 1, with the container type values (Community Registered Value) as per Section 5.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0001             |R C 0 0 0 0 0 0|   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Registered Community Value                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source AS Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Context AS Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Wide Community Target(s) TLV (optional)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Wide Community Exclude Target(s) TLV (optional)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Wide Community Parameter(s) TLV (optional)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Wide BGP Community Type 1

Description format:

TYPE:
    0x0001 (constant for this registry)
FLAGS "F":
    R - Registered bit (Set to 1 for registered values)
    C - Confederation bit (Set when applicable)
HOP COUNT "H":
    Defines domain or sub-domain propagation radius
LENGTH "L":
    Length of the Container Type 1 in octets
REGISTERED COMMUNITY VALUE "R":
    Value of the community in registry
SOURCE AS "S":
    Originator AS of Wide BGP Community
CONTEXT AS "C":
    For registered communities carries predefined meaning
    or otherwise should be set to 0x00000000
TARGET TLV "T":
    Set of atoms containing targets for execution
EXCLUDE TARGET TLV "E":
    Set of atoms containing excluded targets for execution
PARAMETER TLV "P":
    Set of atoms containing optional parameters for execution

BLACKHOLE

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - All transit traffic to destinations for which advertised routes carry such community value should be dropped. It is recommended that specified Autonomous System number should be eligible and verified by BGP Origin Validation functionality to advertise given BGP destinations.

SOURCE FILTER

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - All transit traffic which source addresses have been tagged by such Wide BGP Community should be dropped.

SOURCE DO RPF

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - All transit traffic which source addresses have been tagged by such Wide BGP Community should be subject to Reverse Path Forwarding check when crossing Autonomous System boundaries. Source Autonomous System number specified in the body of this community should directly indicate the peering interfaces on which such RPF check should be performed.

HIGH PRIORITY PREFIX

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - BGP prefixes carrying such Wide BGP Community should be advertised to restarting peers before other prefixes received by given BGP speaker.

ATTACK TARGET

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - The ATTACK_TARGET Registered Wide BGP Community indicates that BGP prefixes carrying such community are receiving unusual amount of unwanted traffic most likely due to some form of network attack. Network devices capable of analyzing and mitigating such attacks can use such community as a hint on what destinations to focus the most.

2.2.2. Advertisement control Registered Wide BGP Communities

NO ADVERTISE TO AS

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (Peer_AS)
    L = 25 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - All routes received which carry such Wide BGP Community containing this value MUST NOT be advertised to BGP peer which Autonomous System number has been listed in the TARGET TLV field of this community.

Semantically specifying the reserved Autonomous System value of 0xFFFFFFFF (ANY AS) would be an equivalent of using NO_ADVERTISE Well-Known Standard BGP Community Attribute.

ADVERTISE TO AS

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (Peer_AS)
    L = 25 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - All routes received carrying such Wide BGP Community containing this value MUST ONLY be advertised to BGP peers which Autonomous System number is specified in the TARGET TLV field of this community.

Semantically specifying the reserved Autonomous System value of 0xFFFFFFFF (ANY AS) would be an equivalent of advertisement to all neighbors. Post execution this community MUST be removed.

ADVERTISE AND SET NO EXPORT

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (Peer_AS)
    L = 25 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - All routes received carrying such Wide BGP Community containing this value MUST be advertised to BGP peer which Autonomous System number is specified in the TARGET TLV field of this community with NO_EXPORT Standard BGP Community attached.

Semantically specifying in TARGET TLV the reserved Autonomous System value of 0xFFFFFFFF (ANY AS) would be an equivalent of advertisement to all neighbors with NO_EXPORT community being set. Post execution this community MUST be removed.

2.2.3. AS source marking Registered Wide BGP Communities

FROM PEER

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System may attach this community to routes received from their EBGP peers to later, when advertising them outside the domain, apply or relax local policies only on such group of destinations.

FROM CUSTOMER

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System may attach this community to routes received from their customers to later, when advertising them outside the domain, apply or relax local policies only on such group of destinations.

INTERNAL

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System may attach this community to routes originated in their own domain to later, when advertising them outside the domain, apply or relax local policies only on such group of destinations.

FROM UPSTREAM

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System may attach this community to routes received from their EBGP upstream peers to later, when advertising them outside the domain, apply or relax local policies only on such group of destinations.

FROM IX

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = none
    L = 18 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System may attach this community to routes received from their EBGP peering sessions with the Internet Exchange peers or with Route Server to later, when advertising them outside the domain, apply or relax local policies only on such group of destinations.

LEARNED FROM AS

```
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = Type_1 (Peer_AS)
    L = 25 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System may attach this community to routes received from their EBGP peer by explicitly tagging them with their peer’s Autonomous System number as a value of the TARGET TLV field. If the AS number is a two octet number first two octest will be

filled with zero. It is possible to use this to also carry private AS number of customers.

2.2.4. Return path influencing Registered Wide BGP Communities

```
PATH HINT
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (AS#)
    L = 25 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System receiving such Wide BGP Community value should prefer for BGP prefixes received with such community (for example by increasing value of local preference on ingress), a BGP path which traverses Autonomous System number which has been specified in the TARGET TLV field of this community. Post execution this community SHOULD be kept.

```
NEGATIVE PATH HINT
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (AS#)
    L = 25 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - Autonomous System receiving such Wide BGP Community value should prefer for BGP prefixes received with such community (for example by increasing value of local preference on ingress), a BGP path which DOES NOT traverses Autonomous System number which has been specified in the TARGET TLV field of this community. Post execution this community SHOULD be kept.

2.2.5. AS_PATH modifying Registered Wide BGP Communities

```
PREPEND N TIMES BY AS
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (AS#)
    L = 29 octets               E = none
    R = IANA assigned           P = Type_4 (0xAA)
```

DESCRIPTION - The Autonomous System specified in the TARGET TLV field of such community should prepend N times (encoded as 0xAA) its own Autonomous System number when advertising routes tagged with this community to peers. Number of requested AS prepends is provided in the PARAMETERS TLV field value. Post execution this community MUST be removed.

```
PREPEND N TIMES TO AS
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (AS#)
    L = 29 octets               E = none
    R = IANA assigned           P = Type_4 (0xAA)
```

DESCRIPTION - The Autonomous System advertising routes externally should prepend N times (encoded as 0xAA) its own Autonomous System number when advertising routes tagged with this community to peer which AS number is defined by TARGET TLV field. Number of requested AS prepends is provided in the PARAMETERS TLV field. Post execution this community MUST be removed.

```
REPLACE BY
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = Type_1 (AS#)
    L = 25 octets               E = none
    R = IANA assigned           P = none
```

DESCRIPTION - All routes marked with such community advertised by an Autonomous System to all of its external peers should have any occurrence of an Autonomous System number specified in the TARGET TLV field replaced with advertising domain's local Autonomous System number. Post execution this community MUST be removed.

2.2.6. Local Preference Registered Community

```
LOCAL PREFERENCE
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = none
    L = 22 octets               E = none
    R = IANA assigned           P = Type_4 (ABBBBBBB)

    SEMANTICS of PARAMETERS TLV
        1 octet 1st bit indicates:
            0-increment, 1-decrement
            7 bits - value of local preference value 1..127
```

DESCRIPTION - Autonomous System may suggest to its EBGP neighbor the following adjustments to the value of local preference as specified by given domain's local policy. The values of requested increment or decrement of local preference value is carried in the PARAMETERS TLV field. Post execution this community MUST be removed.

2.2.7. AS_PATH TTL Registered Community

```
AS_PATH TTL MAX RADIUS
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = none
    L = 22 octets               E = none
    R = IANA assigned           P = Type_4 (0xAA) max AS_PATH radius
```

DESCRIPTION - Autonomous System may suggest to drop advertised prefix by any transit network if its AS_PATH attribute length would be equal or greater to encoded value both inbound or outbound of EBGP session. The value of max AS_PATH length allowed is specified in the PARAMETERS TLV field of the community. Post comparison this community MUST be kept.

2.2.8. GEO-LOCATION Registered Community

```
GEOGRAPHIC LOCATION WHERE BGP ROUTE IS INTRODUCED TO AS
    Type: 0x0001                S = src AS #
    F = 0x80                    C = 0x00000000
    H = Operator's defined      T = none
    L = 26 octets               E = none
    R = IANA assigned           P = Type_8 (5 UTF-8 characters)
```

DESCRIPTION - Autonomous Systems may attach this community to routes received from EBGP neighbors or introduced to BGP by other routing protocols to identify the geographic location where the route was introduced to the AS. The "right-most" two octets of PARAMETERS TLV correspond to an ISO3166-1 alpha-2 country identifier, while the "left-most" three octets may express a more specific geographic location, such as a city or IXP encoded in 3 octets.

Example:

```
Wide BGP Community describing route learnt by the AS at London, GB
     HOP COUNT - operator defined
     LENGTH - 26
     PARAMETERS - 3 octets locality string + 2 octets country id.
```

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0001             |       0x80    |      0xFF     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x001A             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IANA assigned                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           0x000000AS                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           0x00000000                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x08     |            0x0008             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       "L"     |       "O"     |       "N"     |      "G"      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       "B"     |
+-+-+-+-+-+-+-+-+
```

3. Example

Customer of the source AS number 100 requests to execute AS_PATH prepend 4 times when advertising the prefixes to AS number 2424. We will use the following community assigned on ingress or at the prefix origination.

```
PREPEND N TIMES TO AS
    Type: 0x0001                S = 0x00000064 (dec 100)
    F = 0x80                    C = 0x00000000
    H = 0x00                    T = 0x00000978 (dec 2424)
    L = 0x001D (dec 29 octets)  E = none
    R = IANA assigned           P = 0x04 (dec 4)
```

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0001             |       0x80    |      0x00     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x001D             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IANA assigned                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           0x00000064                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           0x00000000                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x01     |            0x0007             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           0x00000978                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x04     |            0x0004             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x04     |
+-+-+-+-+-+-+-+-+
```

4. Security considerations

All the security considerations for BGP Communities as well as for BGP Extended Communities RFCs apply here.

5. IANA Considerations

This document requests IANA to define and maintain a new registry named: "Registered Wide BGP Communities Values". The reserved pool of 0x00000000-0xFFFFFFFF has been defined for its allocations. The allocation policy is on a first come first served basis.

This document makes the following assignments for the Registered Wide BGP Community values:

Name	Type Value
BLACKHOLE	1
SOURCE FILTER	2
SOURCE DO RPF	3
HIGH PRIORITY PREFIX	4
ATTACK TARGET	5
NO ADVERTISE TO AS	6
ADVERTISE TO AS	7
ADVERTISE AND SET NO EXPORT	8
FROM PEER	9
FROM CUSTOMER	10
INTERNAL	11
FROM UPSTREAM	12
FROM IX	13
LEARNED FROM AS	14
PATH HINT	15
PATH NEGATIVE HINT	16
PREPEND N TIMES BY AS	17
PREPEND N TIMES TO AS	18
REPLACE BY	19
LOCAL PREFERENCE	20
AS_PATH TTL MAX RADIUS	21
GEO-LOCATION	22
FREE POOL	23..

6. Contributors

The following people contributed significantly to the content of the document:

Bruno Decraene
France Telecom
38-40 rue du General Leclerc
92794 Issi Moulineaux cedex 9
France
Email: bruno.decraene@orange-ftgroup.com

Shintaro Kojima
OTEMACHI 1st. SQUARE EAST TOWER, 3F
1-5-1, Otemachi,
Chiyoda-ku, Tokyo 100-0004
Japan
Email: koji@mfeed.ad.jp

Juan Alcaide
Cisco Systems
Research Triangle Park, NC
United States
Email: jalcaide@cisco.com

Burjiz Pithawala
Cisco Systems
170 West Tasman Dr
San Jose, CA
United States
Email: bpithaw@cisco.com

Saku Ytti
TDC Oy
Mechelininkatu 1a
00094 TDC
Finland
Email: ytti@tdc.net

Paul Jakma
School of Computing Science, Uni. of Glasgow
Sir Alwyn Williams Building
University of Glasgow
Glasgow
G1 5AE
UK
Email: paulj@dcs.gla.ac.uk

7. Acknowledgments

Authors would like to thank Enke Chen, Pedro Marques, Alton Lo and Jeff Wheeler for their valuable input.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.

8.2. Informative References

[RFC1997] Chandrasekeran, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, August 1996.

[RFC1998] Chen, E. and T. Bates, "An Application of the BGP Community Attribute in Multi-home Routing", RFC 1998, August 1996.

[RFC4384] Meyer, D., "BGP Communities for Data Collection", BCP 114, RFC 4384, February 2006.

[RFC4893] Vohra, Q. and E. Chen, "BGP Support for Four-octet AS Number Space", RFC 4893, May 2007.

[RFC5668] Rekhter, Y., Sangli, S., and D. Tappan, "4-Octet AS Specific BGP Extended Community", RFC 5668, October 2009.

Authors' Addresses

Robert Raszuk (editor)
Mirantis Inc.
615 National Ave. #100
Mt View, CA 94043
USA

Email: robert@raszuk.net

Jeffrey Haas
Juniper Networks
1194 N.Mathilda Ave
Sunnyvale, CA 94089
US

Email: jhaas@pfrc.org

IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2015

R. Raszuk, Ed.
Mirantis Inc.
J. Haas, Ed.
Juniper Networks
A. Lange, Ed.
Alcatel-Lucent
S. Amante
Apple, Inc.
B. Decraene
Orange
P. Jakma
Uni. of Glasgow
R. Steenbergen
nLayer Communications, Inc.
March 6, 2015

Wide BGP Communities Attribute
draft-raszuk-wide-bgp-communities-05

Abstract

Route tagging plays an important role in external BGP [RFC4271] relations, in communicating various routing policies between peers. It is also a very common best practice among operators to propagate various additional information about routes intra-domain. The most common tool used today to attach various information about routes is through the use of BGP communities [RFC1997].

Such information is important to allow BGP speakers to perform some mutually agreed actions without the need to maintain a separate offline database for each tuple of prefix and associated set of action entries.

This document defines a new encoding which will enhance and simplify what can be accomplished today with the use of BGP communities. The most important addition this specification makes over currently defined BGP communities is the ability to specify, carry as well as use for execution an operator’s defined set of parameters. It also provides an extensible platform for any new community encoding needs in the future.

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 RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Protocol Summary 4
2. Wide Community Atoms 5
  2.1. The Autonomous System number list atom 6
  2.2. The IPv4 and IPv6 prefix list atoms 6
  2.3. The Integer list atom 6
  2.4. The IEEE Floating Point Number list atom 7
  2.5. The Neighbor Class list atom 7
  2.6. The User-defined Class list atom 7
  2.7. The UTF-8 String atom 8
3. Wide BGP Community Attribute 8
  3.1. Wide BGP Community Attribute Container Header 8
4. Container Type 1: Wide Community 10
  4.1. Community Value 10

4.2. Source AS Number 10
4.3. Context AS Number 10
4.4. Wide Community Target(s) TLV 11
4.5. Wide Community Exclude Target(s) TLV 12
4.6. Wide Community Parameter(s) TLV 12
4.7. Usage 13
5. The AS-4 List Generic Wide BGP Community 13
5.1. Definition 13
6. Well Known Standard BGP Communities 13
7. Operational Considerations 14
8. Error Handling 14
9. Example 14
9.1. Example Wide Community Definition 14
9.2. Example Wide Community Encoding 15
10. Security considerations 17
11. IANA Considerations 17
12. Change History 18
13. Outstanding Issues 19
14. Contributors 20
15. Acknowledgments 20
16. References 20
16.1. Normative References 20
16.2. Informative References 21
Authors' Addresses 21

1. Introduction

RFC 1997 [RFC1997] defines the BGP Community Attribute. This attribute is used as a tool to carry additional information in BGP routes which may help to automate peering administration. The BGP Communities Attribute consists of one or more sets of four octet values, where each specifies a different community. Except for two reserved ranges, the encoding of community values mandates that the first two octets are to contain the Autonomous System number, with the next two octets containing some locally defined value.

With the introduction of 4-octet Autonomous System numbers by RFC 4893 [RFC4893] it became obvious that BGP Communities as specified in RFC 1997 will not be able to accommodate new AS encoding. In fact RFC 4893 explicitly recommends use of four octets AS specific [RFC5668] extended communities [RFC4360] as a way to encode new 4 octet AS numbers.

While the encoding of 4 octet AS numbers is being addressed by [I-D.ietf-idr-as4octet-extcomm-generic-subtype], neither the base BGP communities (standard or extended) nor as4octet-extcomm-generic document define a sufficient level of encoding freedom which could be of practical use. The authors believe that defining a new BGP Path

Attribute, with the ability to contain locally defined parameters will enhance the current level of network policies, as well as simplify BGP policy management. The proposed simple encoding will also facilitate the delivery of new network services without a need to define a new BGP extension each time.

When defining any new type of tool there is always a unique opportunity to specify a subset of well recognized behaviors. Lists of the current most commonly used BGP communities, as well as provision for a new registry for future definitions will be contained in a separate document.

1.1. Protocol Summary

Each Wide BGP Community consists of three main parts:

1. A Container Header. This header is defined in Section 4. The Container Header encodes:

   * Container Type. One Type is defined in this document.
   * Flags to control behavior.
   * Hop Count to control the degree of Community propagation
   * Length

2. The Community itself, defined as a type of Container. The Type 1 Wide Community is defined in Section 4. The Type 1 Wide Community contains:

   * Community Value: This section defines the action that an operator wishes a router to take.
   * Source AS: This is the AS originating the community.
   * Context AS: This its the AS context from which community should be interpreted.
   * Target(s): This is an optional list that encodes where the community's action should be taken.
   * Exclude Target(s): This is an optional list that encodes where the community's actions should not be taken.
   * Parameters: This is an optional list that encodes additional information that the community's action needs to execute properly.

3. Community Atoms. These are values and lists of values that are common across community actions. They are defined in Section 2.

2. Wide Community Atoms

Wide BGP communities will act on and hence need to encode some distinct atoms of data. These are encoded as TLVs, where each TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|     Type      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Value (variable)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field contains a value of 1-254. The values 0 and 255 are reserved for future use. The TLV types are to be assigned and maintained by IANA registry.

The Length represents the total length of a given TLV's value field in octets.

The Value field contains the TLV value.

Supported format of the TLVs can be:

```
Type  1: Autonomous System number list
Type  2: IPv4 prefix (1 octet prefix length + prefix) list
Type  3: IPv6 prefix (1 octet prefix length + prefix) list
Type  4: Integer list
Type  5: IEEE Floating Point Number list
Type  6: Neighbor Class list
Type  7: User-defined Class list
Type  8: UTF-8 String
```

The semantics of a given atom will depend upon the context in which it is used, as defined by the containing wide community.

In the following sections defining the different atoms, validation rules for the Length of the atom will be presented. If the Length of the atom does not match the rules for that atom, it SHALL be considered malformed. (See Section 8.)

2.1. The Autonomous System number list atom

This atom represents a list of Autonomous System numbers, each 4 octets in size. The minimum Length of this atom is 4 octets. The Length MUST be a multiple of 4.

For consistent treatment, all AS numbers MUST be encoded as 4 octet values. When encoding two octet ASes, the first two octets of this four octet value MUST be filled with zeros.

Two special values are reserved for the Autonomous System atoms:

```
0x00000000 - to indicate "No Autonomous Systems".
0xFFFFFFFF - to indicate "All Autonomous Systems".
```

2.2. The IPv4 and IPv6 prefix list atoms

This atom represents a list of IPv4 or IPv6 prefix. IPv4 and IPv6 prefix atom values are encoded in the same format used by BGP NLRI in Section 4.3 of [RFC4271].

```
+---------------------------+
|  Prefix Length (1 octet)  |
+---------------------------+
|  Prefix (variable)        |
+---------------------------+
```

The Prefix Length for IPv4 prefixes must be in the range of 0..32.

The Prefix Length for IPv6 prefixes must be in the range of 0..128.

The Length field must be able to accommodate the list of prefixes according to the encoding rules. If the Length cannot fully accommodate the required number of octets to encode the Prefix Length and the Prefix, the atom is considered malformed.

2.3. The Integer list atom

This atom represents a list of Integers. Integers are a fixed Length of 4 octets and are stored in network byte order.

The minimum Length of the Integer list atom is 4 octets. The Length MUST be a multiple of 4.

2.4. The IEEE Floating Point Number list atom

This atom represents a list of floating point numbers. Floating point numbers are a fixed Length of 4 octets and are stored in [IEEE.754.1985] format.

This atom represents a list of floating point numbers.

The minimum Length of the Floating Point Number list atom is 4 octets. The Length MUST be a multiple of 4.

2.5. The Neighbor Class list atom

The Neighbor Class list atom represents a classification of a BGP peering session, each 4 octets in size. This class currently can contain three values:

```
1 - Peer:     This class is typically applied to sessions where a
              transit-free relationship exists between two
              providers.
2 - Customer: This class is typically applied to sessions where
              the remote end of the session is operated by a
              customer.
3 - Upstream: This class is typically applied to sessions where the
              remote end of the session is operated by a network
              from which you receive transit routes.
```

The Neighbor Class list atom represents a classification of a BGP peering session.

The minimum Length of the Neighbor Class list atom is 4 octets. The Length MUST be a multiple of 4.

2.6. The User-defined Class list atom

Similar to the Neighbor Class atom, the User-defined Class list atom represents a classification of a network property. The exact property definition is up to the semantics of the defining Autonomous System. The semantics governing a given User-defined Class list are defined by the Context AS Number.

Examples of User-defined Class properties include geography (East, West), continent (North America, Asia, Europe), etc. Similar to the [RFC1997] BGP Communities, it is necessary that the Context AS provide a registry of the value and the semantics of a given community.

The minimum Length of the User-defined Class list atom is 4 octets. The Length of this atom MUST be a multiple of 4.

2.7. The UTF-8 String atom

The UTF-8 String atom represents an arbitrary Unicode string in UTF-8 [RFC3629] format. The Length is required to be of sufficient size to carry the UTF-8 string in the Value field.

Implementations MUST be prepared for truncated/improperly formed UTF-8 strings. When detecting such a string, the implementation should remove trailing octets of a multi-octet sequence in order to have a well-formed string.

Implementations MUST be prepared to receive empty (zero-Length) UTF-8 String atoms as they may be used as Parameters.

3. Wide BGP Community Attribute

This document defines a new BGP Path Attribute, the Wide BGP Community. The attribute type code is (TBA by IANA).

The Wide BGP Community Attribute is an optional, transitive BGP attribute, and may be present only once in the BGP UPDATE message.

The attribute contains a number of typed containers. Any given container type may appear multiple times, unless that container type's definition specifies otherwise.

3.1. Wide BGP Community Attribute Container Header

Containers always start with the following header:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |     Flags     |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This document defines one Type code - Type 1. See the Section 11 for information on additional type registration policies.

Bit	Value	Meaning
0	0	Local community value.
	1	Registered community value.
1	0	Do not decrement Hop Count field across confederation boundaries.
	1	Decrement Hop Count field across confederation boundaries.
2..7	-	MUST be zero when sent and ignored upon receipt.

Flags are defined globally, to apply to all wide community container types.

Table 1: Flags

Bit 0 (aka R bit) set (value 1) indicates that the given container carries a Wide BGP Community which is registered with IANA. When not set (value 0) it indicates that community value which follows is locally assigned with a local meaning.

Bit 1 (aka C bit) is used to manage the propagation scope of a given Wide BGP Community across confederation boundaries. When not set (value of 0), the Hop Count field is not considered at the sub-AS boundaries. When set (value of 1), sub-AS border router follows the same procedure regarding the handling of the Hop Count field as applicable to ASBR at the domain boundary.

The Hop Count field represents the forwarding radius, in units of AS hops, for the given Wide BGP Community. A Hop Count value of zero indicates that this wide community must not cross any further AS boundaries. At each AS boundary, when propagating a given wide community over an EBGP session, the Hop Count field MUST be decremented by the sending EBGP speaker.

The exact same decrement procedures described above apply also to sub-confederation boundaries when the bit 1's flag is set to 1.

The special value of 0xFF indicates that the enclosed community may always be propagated over an EBGP boundary. A Hop Count value of 0xFF MUST NOT be decremented during propagation.

The Length field represents the total length of a given container in octets.

4. Container Type 1: Wide Community

The Wide BGP Community Type 1 container is of variable size (but minimum length 12) and is encoded as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Registered/Local Community Value                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source AS Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Context AS Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Wide Community Target(s) TLV (optional)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Wide Community Exclude Target(s) TLV (optional)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Wide Community Parameter(s) TLV (optional)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Wide BGP Community Type 1

4.1. Community Value

Community Value: 4 octets

The Wide BGP Community value indicates what set of actions a router is requested to take upon reception of a route containing this community. The semantics of this value depend on whether this is a private/local community or registered.

4.2. Source AS Number

Source Autonomous System Number: 4 octets

The Autonomous System number which indicates the originator of this Wide BGP Community.

When the Autonomous System is a two octet number the first two octets of this 4 octet value MUST be filled with zeros.

4.3. Context AS Number

Context Autonomous System Number: 4 octets

The Autonomous System number that indicates the context of the Registered/Local Value. For Wide Community Containers that are Locally defined, the Context Autonomous System number provides the context for the Community Value and thus the meaning of the community as well as any User-defined Classes for that Context AS.

For Registered Community Containers, the Context Autonomous System number MAY be utilized to provide specific meaning for that Registered community. When no such context is implied, this field MUST be 0.

When the wide community is locally registered, the Context Autonomous System Number indicates the AS that defines the format of this wide community for the given Local Value. (In other words, value 1 will likely refer to different formats for AS 1 vs. AS 2.)

4.4. Wide Community Target(s) TLV

The Wide Community Target(s) TLV (Type 1) contains a list of a Wide Community atoms.

Wide Community Targets define the matching criteria for the community. A given wide community may have a number of targets that it applies to. The semantics of these targets will vary on a per community basis. Depending on the definition of the community, targets may be optional.

The value field of the Wide Community Target(s) TLV is a series of Wide Community Atom TLVs. The semantics of any given atom TLV MUST be part of the definition of a given Wide Community.

Typically, Wide Community Targets consist of a series of atoms that have "match any" semantics. Thus, if any given target matches per the semantics of that atom for the community, the community is considered to match and the action defined by the community should be executed.

When no Target(s) TLV is specified, it is considered "match all".

If the semantics of a given atom is undefined for the community in question, it MUST be ignored.

When no targets are required by the definition of a given Wide Community, the Wide Community Target(s) TLV SHOULD NOT be encoded in the community. Implementations MUST be prepared to accept a Wide Community Target(s) TLV with an empty value field.

4.5. Wide Community Exclude Target(s) TLV

The Wide Community Exclude Target(s) TLV (Type 2) contains a list of a Wide Community atoms.

Wide Community Exclude Targets define criteria by which the community is considered to NOT match. Depending on the semantics of the Wide Community, Exclude Target(s) may be optional.

The semantic of the Wide Community Exclude Target(s) is to match all specified Target(s) with the exception of those listed in this TLV.

The value field of the Wide Community Exclude Target(s) TLV is a series of Wide Community Atom TLVs. The semantics of any given atom TLV MUST be part of the definition of a given Wide Community.

If the semantics of a given atom is undefined for the community in question, it MUST be ignored.

If the Wide Community Target(s) TLV and the Wide Community Exclude Target(s) TLV have conflicting semantics, priority MUST be given to the Wide Community Exclude Target(s) TLV.

When no exclude targets are required by the definition of a given Wide Community, the Wide Community Exclude Target(s) TLV SHOULD NOT be encoded in the community. Implementations MUST be prepared to accept a Wide Community Exclude Target(s) TLV with an empty value field.

4.6. Wide Community Parameter(s) TLV

The Wide Community Parameter(s) TLV (Type 3) contains a list of a Wide Community atoms.

A given wide community may have parameters which are used as inputs for executing actions defined for that community. These parameters, and any constraints implied by the parameters, MUST be defined by the wide community definition. Parameters consist of an ordered set of atom sub-TLVs. The semantics of any specific positional instance of an atom MUST be defined by the wide community.

If it is the case that a parameter for a given community is of an unexpected type or length, the community MUST be ignored.

If it is the case that there are too many or two few parameters for a given community, the community MUST be ignored.

When no parameters are required by the definition of a given Wide Community, the Wide Community Paramters TLV SHOULD NOT be encoded in the community. Implementations MUST be prepared to accept a Wide Community Parameter TLV with an empty value field.

4.7. Usage

The detailed interpretation of the targets or parameters SHALL be provided when describing given community type in a separate document or when locally defined by an operator.

5. The AS-4 List Generic Wide BGP Community

In standard [RFC1997] BGP Communities, a commonly deployed format is AS:0x0000-0xFFFF. The left-hand-side AS is the context AS while the right-hand-side represents an action intended to be taken upon that AS. While the [I-D.ietf-idr-as4octet-extcomm-generic-subtype] Extended Community format is intended to address the use cases where the right-hand-side represents a semi-opaque integer value, operators that desire to utilize the right-hand-side as an Autonomous System Number have been unable to do so.

In the AS:0x0000-0xFFFF format, there are no explicit semantics on the action for a given right-hand-side. The semantics are implicit to the Autonomous System number itself. While such a format doesn't take advantage of the more flexible semantics of Wide BGP Communities, the format is capable of addressing this operational need.

This document defines a Registered Wide BGP Community to address that need.

5.1. Definition

```
Community Value: 1 - AS-4 List Generic Wide BGP Community
Source AS Number: <Your AS number>
Context AS Number: <Targeted AS number>
Wide Community Target(s): <MUST BE ABSENT>
Wide Community Exclude Target(s): <MUST BE ABSENT>
Wide Community Parameter(s): <Wide Community Atom Type 4>
```

6. Well Known Standard BGP Communities

According to RFC 1997, as well as IANA's Well-Known BGP Communities registry, the following BGP communities are defined to have global significance:

0xFFFF0000	planned-shut	[draft-francois-bgp-gshut]
0xFFFFFF01	NO_EXPORT	[RFC1997]
0xFFFFFF02	NO_ADVERTISE	[RFC1997]
0xFFFFFF03	NO_EXPORT_SUBCONFED	[RFC1997]
0xFFFFFF04	NOPEER	[RFC3765]

This document recommends for simplicity as well as for avoidance of backward compatibility issues the continued use of BGP Standard Community Attribute type 8 as defined in RFC 1997 to distribute non Autonomous System specific Well-Known BGP Communities.

For the same reason, this document does not intended to obsolete the currently defined and deployed BGP Extended Communities.

7. Operational Considerations

Having two different ways to propagate locally assigned BGP communities, one via the use of Standard BGP Communities and the other one via the use of Wide BGP Communities, may seem to potentially cause problems when considering propagation of conflicting actions. However, even at present, an operator may append Standard BGP Communities with conflicting information. It is therefore recommended that any implementation, in supporting both standard and Wide BGP communities, allow for their easy inbound and outbound processing. The actual execution of all communities should be treated as a union and, if supported by an implementation, their execution permissions are to be a local configuration matter.

8. Error Handling

If any atom in a Wide Community container’s Exclude Targets TLV is unrecognized, no actions should be taken for that Wide Community. While the Targets TLV is meant to be inclusive, the Exclude Targets TLV is meant to be proscriptive of applying the action.

If any Wide Community container or atoms contained therein are determined to be malformed, the Wide Community Path attribute must be considered malformed. BGP implementations should use "treat-as-withdraw" semantics as defined in [I-D.ietf-idr-error-handling].

9. Example

9.1. Example Wide Community Definition

An operator of an AS 64496, wishes to locally define a Wide Community with the semantics of permitting AS_PATH prepending with targets that include AS numbers of peer ASes and peers who have been marked with a set of enumerated city locations.

AS 64496 has established a registered set of values to use for its User-defined Class:

100 - Amsterdam
101 - New York
102 - San Francisco
103 - Tokyo
104 - Moscow

Target semantics:

The Autonomous System Number list atom refers to the target peer AS Numbers.

The User-defined Class for AS 64496 has been defined elsewhere and the values 100..104 may be used for this locally defined Wide Community.

The Targets TLV MUST contain at least one entry.

The Exclude Targets TLV MAY contain entries of the above supported atoms.

The semantics of all other atoms are undefined for this community.

Parameter semantics:

The parameter TLV shall consist of exactly one Integer atom value that is constrained to have a value of 2..8.

9.2. Example Wide Community Encoding

AS_PATH prepend 4 TIMES TO AS 2424, AS 8888, to peers marked as Amsterdam (100) or to peers marked Moscow (104), but not to peers in New York (101).

Use Hop Count 0 to request the receiving router to not propagate this wide community.

Locally community value (flag bit 0 = 0).
Do not decrement Hop Count field across confederation boundaries (0)

Local community 1 for sample AS 64496.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Container Type 1 (1)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0 0|
+-+-+-+-+-+-+-+-+
| Hop Count: 0  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Length:                     57 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Community: LOCAL PREPEND ACTION CATEGORY                        1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Own ASN                                                     64496 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Context ASN#                                                64496 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Target TLV (1)|   Length:                                     22 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ASN List (2)  |   Length:                                      8 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Target ASN#                                                  2424 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Target ASN#                                                  8888 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  User List(7) |   Length:                                      8 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Amsterdam                                                     100 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Moscow                                                        104 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|ExcTargetTLV(2)|   Length:                                      7 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  User List(7) |   Length:                                      4 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| New York                                                      101 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Param TLV (3) |   Length:                                      7 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Integer  (4) |   Length:                                      4 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Prepend #                                                       4 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

10. Security considerations

All the security considerations for BGP Communities as well as for BGP RFCs apply here.

Given the flexibility and power offered by Wide BGP communities, it is important to consider the additional possibilities allowed by their definition. In particular, for locally defined Wide BGP Communities, it may be wise to restrict the range of parameters. For registred Wide BGP Communities, the security considerations of the document defining them MUST address issues specific to those newly defined Wide Communities.

Security considerations specific to Wide BGP Communities will be discussed in a later revision of this draft.

11. IANA Considerations

This document defines a new BGP Path Attribute called Wide BGP Community Attribute. For this new type IANA is to allocate a new value in the corresponding registry:

Registry Name: BGP Path Attributes

This document makes the following assignments for the optional, transitive Wide BGP Communities Attribute:

Name	Type Value
Wide BGP Community Attribute	TBA

This document requests IANA to define and maintain a new registry named: "Wide BGP Communities Attribute Container Types".

The pool of: 0x0000-0xFFFF has been defined for its allocations. The allocation policy is on a first come, first served basis.

This document makes the following assignments for the Wide BGP Communities Container Types values:

Name	Type Value
Reserved	0x0000
Type 1, Wide BGP Community	0x0001
Types 2-1023 to be allocated using IETF Consensus	
Types 1024-64511 to be allocated first come, first served	
Types 64512-65534 are reserved for experimental use	
Reserved	0xFFFF

This document requests IANA to define and maintain a new registry named: "Wide BGP Communities Atom Types". The pool of 0x0000-0xFFFF has been defined for its allocations.

This document makes the following assignments for the Wide BGP Communities Atom Type values:

Name	Type Value
Reserved	0x00
Autonomous System Number List	0x01
IPv4 Prefix list	0x02
IPv6 Prefix list	0x03
Integer list	0x04
IEEE Floating Point Number list	0x05
Neighbor Class list	0x06
User-defined Class list	0x07
UTF-8 string	0x08
Reserved	0xFF

This document requests IANA to define and maintain a new registry named: "Registered Wide BGP Communities". The pool of 0x00000000-0FFFFFFFF has ben defined for its allocation.

This document makes the following assignments for the Registered Wide BGP Communities:

Name	Type Value
Reserved	0x00
AS-4 List Generic Wide BGP Community	0x01
Reserved	0xFFFFFFFF

Values 2-1023 are to be allocated using IETF Consensus. Values 64512-65534 are reserved for experimental use. All other values are available on a first-come, first served basis.

12. Change History

Changes from -03 via -04 to -05:

Update the Introduction.

Substantial re-work of atom types removing proposed Group container and moving atoms to be lists.

Added the Exclude Targets TLV to the Wide Community container.

Added a section on error handling.

Updated the example.

Changes from -02 to -03:

Removed C and R named bit fields originally from -00.

Rename Target AS field to Context AS.

Make Integer Atom a fixed 4 octets in length.

Add Neighbor Class Atom

Rename TTL to Hop Count

Changes from -01 to -02:

The Type field has been expanded to 2 octets.

The Length field has been moved to the common header.

Changed format to use TLVs.

Added atom TLV to define well defined syntactic items.

Added TLVs to distinguish targets from parameters.

Various editorial changes to language.

13. Outstanding Issues

The following are known issues that have yet to be resolved in this draft:

- o The interaction of the Container TTL field with VPN peers.
- o The name Wide Communities is overloaded in this document. The scope of this feature has evolved since the initial -00 of the draft. The general feature of a containerized BGP Community extension and the Type 1 container, the Wide community, currently share names. "There are only two hard things in Computer Science: cache invalidation and naming things."

14. Contributors

The following people contributed significantly to the content of the document:

Shintaro Kojima
OTEMACHI 1st. SQUARE EAST TOWER, 3F
1-5-1, Otemachi,
Chiyoda-ku, Tokyo 100-0004
Japan
Email: koji@mfeed.ad.jp

Juan Alcaide
Cisco Systems
Research Triangle Park, NC
United States
Email: jalcaide@cisco.com

Burjiz Pithawala
Cisco Systems
170 West Tasman Dr
San Jose, CA
United States
Email: bpithaw@cisco.com

Saku Ytti
TDC Oy
Mechelininkatu 1a
00094 TDC
Finland
Email: ytti@tdc.net

15. Acknowledgments

This document owes draft-lange-flexible-bgp-communities a debt for the inspiration of many features contained herein.

The authors would like to thank Enke Chen, Pedro Marques and Alton Lo for their valuable input.

16. References

16.1. Normative References

[I-D.ietf-idr-error-handling]
Scudder, J., Chen, E., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", draft-ietf-idr-error-handling-05 (work in progress), February 2014.

[IEEE.754.1985]
Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

16.2. Informative References

[I-D.ietf-idr-as4octet-extcomm-generic-subtype]
Rao, D., Mohapatra, P., and J. Haas, "Generic Subtype for BGP Four-octet AS specific extended community", draft-ietf-idr-as4octet-extcomm-generic-subtype-06 (work in progress), October 2012.

[RFC1997] Chandrasekeran, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, August 1996.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.

[RFC4893] Vohra, Q. and E. Chen, "BGP Support for Four-octet AS Number Space", RFC 4893, May 2007.

[RFC5668] Rekhter, Y., Sangli, S., and D. Tappan, "4-Octet AS Specific BGP Extended Community", RFC 5668, October 2009.

Authors' Addresses

Robert Raszuk (editor)
Mirantis Inc.
615 National Ave. #100
Mt View, CA 94043
USA

Email: robert@raszuk.net

Jeffrey Haas (editor)
Juniper Networks
1194 N.Mathilda Ave
Sunnyvale, CA 94089
US

Email: jhaas@juniper.net

Andrew Lange (editor)
Alcatel-Lucent
777 E. Middlefield Road
Mountain View, CA 94043
US

Email: andrew.lange@alcatel-lucent.com

Shane Amante
Apple, Inc.
1 Infinite Loop
Cupertino, CA 95014
US

Email: amante@apple.com

Bruno Decraene
Orange
38-40 rue du General Leclerc
Issy Moulineaux cedex 9 92794
France

Email: bruno.decraene@orange.com

Paul Jakma
University of Glasgow
School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
Glasgow G12 8QQ
UK

Email: paulj@dcs.gla.ac.uk

Richard A Steenbergen
nLayer Communications, Inc.
209 W Jackson Blvd
Chicago, IL 60606
US

Email: ras@nlayer.net

Interdomain Routing
Internet-Draft
Intended status: Informational
Expires: December 16, 2015

A. Shaikh, Ed.
Google
R. Shakir, Ed.
BT
K. Patel, Ed.
Cisco
S. Hares, Ed.
Huawei
K. D'Souza
AT&T
D. Bansal
Microsoft
A. Clemm
A. Zhdankin
Cisco
M. Jethanandani
Ciena
X. Liu
Ericsson
June 14, 2015

BGP Model for Service Provider Networks
draft-shaikh-idr-bgp-model-02

Abstract

This document defines a YANG data model for configuring and managing BGP, including protocol, policy, and operational aspects based on data center, carrier and content provider operational requirements.

Status of This Memo

1. Introduction

YANG [RFC6020] is a data modeling language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for a number of interfaces, such as CLIs and programmatic APIs.

This document describes a YANG data model for the BGP [RFC4271] protocol, including various protocol extensions, policy configuration, as well as defining key operational state data. The model is intended to be vendor-neutral, in order to allow operators to manage BGP configuration in heterogeneous environments with routers supplied by multiple vendors. The model is also intended to be readily mapped to existing implementations to facilitate support from as large a set of routing hardware and software vendors as possible.

1.1. Goals and approach

The model covers the base BGP features that are deployed across major implementations and the common BGP configurations in use across a number of operator network deployments. In particular, this model attempts to cover BGP features defined in [RFC4271], [RFC1997], [RFC4456], [RFC4760], [RFC3065], [RFC2439], [RFC4724], and [RFC6811].

Along with configuration of base BGP features, this model also addresses policy configuration, by providing "hooks" for applying policies, and also defining BGP-specific policy features. The BGP policy features are intended to be used with the general routing

policy model defined in [I-D.shaikh-rtgwg-policy-model]. The model also supports operational state data to enable a common model for reading BGP-related state from a BGP speaker.

For the base BGP features, the focus of the model described in this document is on providing configuration and operational state information relating to:

- o The global BGP instance, and neighbors whose configuration is specified individually, or templated with the use of peer-groups.

- o The address families that are supported by peers, and the global configuration which relates to them.

- o The policy configuration "hooks" and BGP-specific policy features that relate to a neighbor - controlling the import and export of NLRIs.

As mentioned earlier, any configuration items that are deemed to be widely available in existing major BGP implementations are included in the model. Additional, more esoteric, configuration items that are not commonly used, or only available from a single implementation, are omitted from the model with an expectation that they will be available in companion modules that augment the current model. This allows clarity in identifying data that is part of the vendor-neutral base model.

Where possible, naming in the model follows conventions used in available standards documents, and otherwise tries to be self-explanatory with sufficient descriptions of the intended behavior. Similarly, configuration data value constraints and default values, where used, are based on recommendations in current standards documentation. Since implementations vary widely in this respect, this version of the model specifies only a limited set of defaults and ranges with the expectation of being more prescriptive in future versions based on actual operator use.

2. Model overview

The BGP model is defined across several YANG modules but at a high level is organized into Six elements:

- o base protocol configuration -- configuration affecting BGP protocol-related operations, defined at various levels of hierarchy.

- o multiprotocol configuration -- configuration affecting individual address-families within BGP [RFC4760].

- o neighbor configuration -- configuration affecting an individual neighbor within BGP.

- o neighbor multiprotocol configuration -- configuration affecting individual address-families for a neighbor within BGP.

- o policy configuration -- hooks for application of the policies defined in [I-D.shaikh-rtgwg-policy-model] that act on routes sent (received) to (from) peers or other routing protocols and BGP-specific policy features.

- o operational state -- variables used for monitoring, management, etc. of BGP operations.

These modules also make use of standard Internet types, such as IP addresses and prefixes, autonomous system numbers, etc., defined in RFC 6991 [RFC6991].

Throughout the model, the approach described in [I-D.openconfig-netmod-opstate] is used to represent configuration (intended state), operational and derived state data. That is to say, that each container holds a "config" and "state" sub-container - with the config container being used for configurable parameters, and the state container container holding representing both the operational state of configurable leaves, and derived counters and statistical information.

2.1. BGP protocol configuration

The BGP protocol configuration model is organized hierarchically, much like the majority of router implementations. That is, configuration items can be specified at multiple levels, as shown below.

```
+--rw bgp!
   +--rw global
   |  +-- (global-configuration-options)
   +--rw neighbors
   |  +--rw neighbor* [neighbor-address]
   |  +-- (neighbor-configuration-options)
   +--rw peer-groups
      +--rw peer-group* [peer-group-name]
      +-- (neighbor-configuration-options)
```

Users may specify configuration at a higher level and have it apply to all lower-level items, or provide overriding configuration at a lower level of the hierarchy. Overriding configuration items are optional, with neighbor specific configuration being the most

specific or lowest level, followed by peer-group, and finally global. Global configuration options reflect a subset of the peer-group or neighbor specific configuration options which are relevant to the entire BGP instance.

The model makes the simplifying assumption that most of the configuration items are available at all levels of the hierarchy. That is, very little configuration is specific to a particular level in the hierarchy, other than obvious items such as "group-name" only being available for the peer group-level config. A notable exception is for sub-address family configuration where some items are only applicable for a given AFI-SAFI combination.

In order to allow common configuration to be applied to a set of neighbors, all neighbor configuration options are available within a peer-group. A neighbor is associated to a particular peer-group through the use of a peer-group leaf (which provides a reference to a configured item in the peer-group list).

Address-family configuration is made available in multiple points within the model - primarily within the global container, where instance-wide configuration can be set (for example, global protocol parameters, the BGP best path route selection options, or global policies relating to the address-family); and on a per-neighbor or per-peer-group basis, where address-families can be enabled or disabled, and policy associated with the parent entity applied. Within the afi-safi container, generic configuration that applies to all address-families (e.g., whether the AFI-SAFI is enabled) is presented at the top-level, with address-family specific containers made available for options relating to only that AFI-SAFI. Within the current revision of the model a generic set of address-families, and common configuration and state options are included - further work is expected to add additional parameters to this area of the model.

The following address-families are currently supported by the model:

```
+--rw bgp!
  +--rw global
     +--rw afi-safis
        +--rw afi-safi* [afi-safi-name]
           +--rw afi-safi-name    -> ../config/afi-safi-name
           |
           +--rw ipv4-unicast
           |     ...
           +--rw ipv6-unicast
           |     ...
           +--rw ipv4-labelled-unicast
           |     ...
           +--rw ipv6-labelled-unicast
           |     ...
           +--rw l3vpn-ipv4-unicast
           |     ...
           +--rw l3vpn-ipv6-unicast
           |     ...
           +--rw l3vpn-ipv4-multicast
           |     ...
           +--rw l3vpn-ipv6-multicast
           |     ...
           +--rw l2vpn-vpls
           |     ...
           +--rw l2vpn-evpn
           |     ...
```

2.2. Policy configuration overview

The BGP policy configuration model references the generic YANG routing policy model described in [I-D.shaikh-rtgwg-policy-model], which represents a condition-action policy framework for routing. This model adds BGP-specific conditions (e.g., matching on the community attribute), and actions (e.g., setting local preference) to the generic policy framework.

Policies that are defined in the routing-policy model are referenced in multiple places within the model:

- o within the global instance, where a policy applies to all address-families for all peers.

- o on a global AFI-SAFI basis, where policies apply to all peers for a particular address-family.

- o on a per-peer-group or per-neighbor basis - where the policy applies to all address-families for the particular parent entity.

o on a per-afi-safi basis within a neighbor or peer-group context, where the policy is specific to the AFI-SAFI for a a specific neighbor or group.

```
+--rw bgp
   +--rw global
   |  +--rw afi-safi
   |  |  +--rw afi-safi* [afi-safi-name]
   |  |     +--rw apply-policy
   |  +--rw apply-policy
   +--rw neighbors
   |  +--rw neighbor* [neighbor-address]
   |     +--rw afi-safi
   |     |  +--rw afi-safi* [afi-safi-name]
   |     |     +--rw apply-policy
   |     +--rw apply-policy
   +--rw peer-groups
      +--rw peer-group* [peer-group-name]
         +--rw afi-safi
         |  +--rw afi-safi* [afi-safi-name]
         |     +--rw apply-policy
         +--rw apply-policy
```

2.3. Operational data overview

The BGP operational model contains a set of parameters which relate to the operational state of the various elements of the BGP router. As noted in Section 2 - the approach described in [I-D.openconfig-netmod-opstate] is utilized for the inclusion of operational and statistical data. To this end, the "_state" groupings (those that contain derived operational parameters) are contained within the BGP operational model - and included within the relevant "state" containers throughout the core BGP model. In some cases, operational information may be relevant to one instance of a common grouping, but not another - for example, the number of received, advertised, and installed prefixes is relevant on a per-neighbor-basis, but is not required (or meaningful) in the peer-group context. To enable state to be added to particular contexts, the tree is augmented through the base BGP module to add these variables, without requiring separate groupings.

3. Security Considerations

BGP configuration has a significant impact on network operations, and as such any related protocol or model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer BGP configuration and operational data. Note that use of alternate transport or data encoding (e.g., JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

Most of the data elements in the configuration model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

4. IANA Considerations

This YANG data model and the component modules currently use a temporary ad-hoc namespace. If and when it is redirected for the standards track, an appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The BGP YANG modules will be registered in the "YANG Module Names" registry [RFC6020].

5. YANG modules

The modules comprising the BGP configuration and operational model are described by the YANG modules in the sections below. The base module imports the other modules to create the overall model.

5.1. BGP base items

```
<CODE BEGINS> file bgp.yang
module bgp {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/bgp";

  prefix "bgp";

  // import some basic inet types
  import ietf-inet-types { prefix inet; }
  import bgp-multiprotocol { prefix bgp-mp; }
  import routing-policy { prefix rpol; }
  import bgp-types { prefix bgp-types; }
  import bgp-operational { prefix bgp-op; }

  // meta
  organization
    "OpenConfig working group";
```

```
     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "This module describes a YANG model for BGP protocol
       configuration.It is a limited subset of all of the configuration
       parameters available in the variety of vendor implementations,
       hence it is expected that it would be augmented with vendor-
       specific configuration data as needed. Additional modules or
       submodules to handle other aspects of BGP configuration,
       including policy, VRFs, VPNs, and additional address families
       are also expected.

       This model supports the following BGP configuration level
       hierarchy:

         BGP
           |
           +-> [ global BGP configuration ]
             +-> AFI / SAFI global
           +-> peer group
             +-> [ peer group config ]
             +-> AFI / SAFI [ per-AFI overrides ]
           +-> neighbor
             +-> [ neighbor config ]
             +-> [ optional pointer to peer-group ]
             +-> AFI / SAFI [ per-AFI overrides ]";

     revision "2015-05-15" {
         description
           "";
         reference "TBD";
     }

     grouping bgp-global_config {
       description
         "Global configuration options for the BGP router.";

       leaf as {
         type inet:as-number;
         mandatory "true";
         description
           "Local autonomous system number of the router.  Uses
           the 32-bit as-number type from the model in RFC 6991.";
       }
       leaf router-id {
         type inet:ipv4-address;
```

```
        description
          "Router id of the router, expressed as an
          32-bit value, IPv4 address.";
      }
    }

    grouping bgp-default-route-distance_config {
      description
        "Configuration options relating to the administrative distance
        (or preference) assigned to routes received from different
        sources (external, internal, and local).";

      leaf external-route-distance {
        type uint8 {
          range "1..255";
        }
        description
          "Administrative distance for routes learned from external
          BGP (eBGP).";
      }
      leaf internal-route-distance {
        type uint8 {
          range "1..255";
        }
        description
          "Administrative distance for routes learned from internal
          BGP (iBGP).";
      }
    }

    grouping bgp-confederation_config {
      description
        "Configuration options specifying parameters when the local
        router is within an autonomous system which is part of a BGP
        confederation.";

      leaf enabled {
        type boolean;
        description
          "When this leaf is set to true it indicates that
          the local-AS is part of a BGP confederation";
      }

      leaf identifier {
        type inet:as-number;
        description
          "Confederation identifier for the autonomous system.";
      }
```

```
      leaf-list member-as {
        type inet:as-number;
        description
          "Remote autonomous systems that are to be treated
          as part of the local confederation.";
      }
    }

    grouping bgp-neighbor_config {
      description
        "Neighbor level configuration items.";

      leaf peer-as {
        type inet:as-number;
        description
          "AS number of the peer.";
      }

      leaf local-as {
        type inet:as-number;
        description
          "The local autonomous system number that is to be used
          when establishing sessions with the remote peer or peer
          group, if this differs from the global BGP router
          autonomous system number.";
      }

      leaf peer-type {
          type bgp-types:peer-type;
          description
            "Explicitly designate the peer or peer group as internal
            (iBGP) or external (eBGP).";
      }

      leaf auth-password {
        type string;
        description
          "Configures an MD5 authentication password for use with
          neighboring devices.";
      }

      leaf remove-private-as {
        // could also make this a container with a flag to enable
        // remove-private and separate option.  here, option implies
        // remove-private is enabled.
        type bgp-types:remove-private-as-option;
        description
          "Remove private AS numbers from updates sent to peers.";
```

```
      }

      leaf route-flap-damping {
        type boolean;
        default false;
        description
          "Enable route flap damping.";
      }

      leaf send-community {
        type bgp-types:community-type;
        default "NONE";
        description
          "Specify which types of community should be sent to the
          neighbor or group. The default is to not send the
          community attribute";
      }

      leaf description {
        type string;
        description
          "An optional textual description (intended primarily for use
          with a peer or group";
      }
    }

    grouping bgp-neighbor-timers_config {
      description
        "Config parameters related to timers associated with the BGP
        peer";

      leaf connect-retry {
        type decimal64 {
          fraction-digits 2;
        }
        default 30;
        description
          "Time interval in seconds between attempts to establish a
          session with the peer.";
      }

      leaf hold-time {
        type decimal64 {
          fraction-digits 2;
        }
        default 90;
        description
          "Time interval in seconds that a BGP session will be
```

```
          considered active in the absence of keepalive or other
          messages from the peer.  The hold-time is typically
          set to 3x the keepalive-interval.";
        reference
          "RFC 4271 - A Border Gateway Protocol 4, Sec. 10";
      }

      leaf keepalive-interval {
        type decimal64 {
          fraction-digits 2;
        }
        default 30;
        description
          "Time interval in seconds between transmission of keepalive
          messages to the neighbor.  Typically set to 1/3 the
          hold-time.";
      }

      leaf minimum-advertisement-interval {
        type decimal64 {
          fraction-digits 2;
        }
        default 30;
        description
          "Minimum time which must elapse between subsequent UPDATE
          messages relating to a common set of NLRI being transmitted
          to a peer. This timer is referred to as
          MinRouteAdvertisementIntervalTimer by RFC 4721 and serves to
          reduce the number of UPDATE messages transmitted when a
          particular set of NLRI exhibit instability.";
        reference
          "RFC 4271 - A Border Gateway Protocol 4, Sec 9.2.1.1";
      }
    }

    grouping bgp-neighbor-transport_config {
      description
        "Configuration parameters relating to the transport protocol
        used by the BGP session to the peer";

      leaf tcp-mss {
        type uint16;
        description
          "Sets the max segment size for BGP TCP sessions.";
      }

      leaf mtu-discovery {
        type boolean;
```

```
        default false;
        description
          "Turns path mtu discovery for BGP TCP sessions on (true)
          or off (false)";
      }

      leaf passive-mode {
        type boolean;
        default false;
        description
          "Wait for peers to issue requests to open a BGP session,
          rather than initiating sessions from the local router.";
      }

      leaf local-address {
        type union {
          type inet:ip-address;
          type string;
        }
        //TODO:  the string should be converted to a leafref type
        //to point to an interface when YANG 1.1 is available with
        //leafrefs in union types.
        description
          "Set the local IP (either IPv4 or IPv6) address to use
          for the session when sending BGP update messages.  This
          may be expressed as either an IP address or reference
          to the name of an interface.";
      }
    }

    grouping bgp-neighbor-error-handling_config {
      description
        "Configuration parameters relating to enhanced error handling
        behaviours for BGP";

      leaf treat-as-withdraw {
        type boolean;
        default "false";
        description
          "Specify whether erroneous UPDATE messages for which the
          NLRI can be extracted are reated as though the NLRI is
          withdrawn - avoiding session reset";
        reference "draft-ietf-idr-error-handling-16";
      }
    }

    grouping bgp-neighbor-logging-options_config {
      description
```

```
        "Configuration parameters specifying the logging behaviour for
        BGP sessions to the peer";

      leaf log-neighbor-state-changes {
        type boolean;
        default "true";
        description
          "Configure logging of peer state changes.  Default is
          to enable logging of peer state changes.";
      }
    }

    grouping bgp-neighbor-multihop_config {
      description
        "Configuration parameters specifying the multihop behaviour for
        BGP sessions to the peer";

      leaf enabled {
        type boolean;
        default "false";
        description
          "When enabled the referenced group or neighbors are permitted
          to be indirectly connected - including cases where the TTL
          can be decremented between the BGP peers";
      }

      leaf multihop-ttl {
        type uint8;
        description
          "Time-to-live value to use when packets are sent to the
          referenced group or neighbors and ebgp-multihop is enabled";
      }
    }

    grouping bgp-neighbor-route-reflector_config {
      description
        "Configuration parameters determining whether the behaviour of
        the local system when acting as a route-reflector";

      leaf route-reflector-cluster-id {
        type bgp-types:rr-cluster-id-type;
        description
          "route-reflector cluster id to use when local router is
          configured as a route reflector.  Commonly set at the group
          level, but allows a different cluster
          id to be set for each neighbor.";
      }
```

```
      leaf route-reflector-client {
        type boolean;
        default "false";
        description
          "Configure the neighbor as a route reflector client.";
      }
    }

    grouping bgp-neighbor-as-path-options_config {
      description
        "Configuration parameters allowing manipulation of the AS_PATH
        attribute";

      leaf allow-own-as {
        type uint8;
        default 0;
        description
          "Specify the number of occurrences of the local BGP speaker's
          AS that can occur within the AS_PATH before it is rejected.";
      }

      leaf replace-peer-as {
        type boolean;
        default "false";
        description
          "Replace occurrences of the peer's AS in the AS_PATH
          with the local autonomous system number";
      }
    }

    grouping bgp-neighbor-add-paths_config {
      description
        "Configuration parameters specfying whether the local system
        will send or receive multiple paths using ADD_PATHS";

      leaf receive {
        type boolean;
        default false;
        description
          "Enable ability to receive multiple path advertisements
          for an NLRI from the neighbor or group";
      }

      leaf send-max {
        type uint8;
        description
          "The maximum number of paths to advertise to neighbors
          for a single NLRI";
```

```
      }
    }

    grouping bgp-neighbor-peer-group_config {
      description
        "Configuration parameters indicating whether the specified peer
        is to be considered as part of a peer-group - and therefore
        inherit its configuration";

      leaf peer-group {
        type leafref {
          // we are at /bgp/neighbors/neighbor/
          path "/bgp/peer-groups/peer-group/peer-group-name";
          require-instance true;
        }
        description
          "The peer-group with which this neighbor is associated";
      }
    }

    grouping bgp-graceful-restart {
      description
        "Configures BGP graceful restart, which is a negotiated
        option that indicates that a BGP speaker is able to retain
        forwarding state when a BGP session restarts";

      reference "RFC 4724: Graceful Restart Mechanism for BGP";
      container graceful-restart {
        description
          "Parameters relating the graceful restart mechanism for BGP";
        container config {
          description
            "Configuration parameters relating to graceful-restart";
          uses bgp-neighbor-graceful-restart_config;
        }
        container state {
          config false;
          description
            "State information associated with graceful-restart";
          uses bgp-neighbor-graceful-restart_config;
        }
      }
    }

    grouping bgp-neighbor-graceful-restart_config {
      description
        "Configuration parameters relating to BGP graceful restart.";
```

```
      leaf enabled {
        type boolean;
        description
          "Enable or disable the graceful-restart capability.";
      }

      leaf restart-time {
        type uint16 {
          range 0..4096;
        }
        description
          "Estimated time (in seconds) for the local BGP speaker to
          restart a session. This value is advertise in the graceful
          restart BGP capability.  This is a 12-bit value, referred to
          as Restart Time in RFC4724.  Per RFC4724, the suggested
          default value is <= the hold-time value.";
      }

      leaf stale-routes-time {
        type decimal64 {
          fraction-digits 2;
        }
        description
          "An upper-bound on the time thate stale routes will be
          retained by a router after a session is restarted. If an
          End-of-RIB (EOR) marker is received prior to this timer
          expiring stale-routes will be flushed upon its receipt - if
          no EOR is received, then when this timer expires stale paths
          will be purged. This timer is referred to as the
          Selection_Deferral_Timer in RFC4724";
      }

      leaf helper-only {
        type boolean;
        description
          "Enable graceful-restart in helper mode only. When this
          leaf is set, the local system does not retain forwarding
          its own state during a restart, but supports procedures
          for the receiving speaker, as defined in RFC4724.";
      }
    }

    // ***********************************************************
    // *              configuration context containers           *
    // ***********************************************************

    grouping bgp-global-base {
      description
```

```
        "Global configuration parameters for the BGP router";

      container config {
        description
          "Configuration parameters relating to the global BGP router";
        uses bgp-global_config;
      }
      container state {
        config false;
        description
          "State information relating to the global BGP router";
        uses bgp-global_config;
        uses bgp-op:bgp-global_state;
      }

      uses bgp-mp:bgp-route-selection-options;

      container default-route-distance {
        description
          "Administrative distance (or preference) assigned to
          routes received from different sources
          (external, internal, and local).";

        container config {
          description
            "Configuration parameters relating to the default route
            distance";
          uses bgp-default-route-distance_config;
        }
        container state {
          config false;
          description
            "State information relating to the default route distance";
          uses bgp-default-route-distance_config;
        }
      }

      container confederation {
        description
          "Parameters indicating whether the local system acts as part
          of a BGP confederation";

        container config {
          description
            "Configuration parameters relating to BGP confederations";
          uses bgp-confederation_config;
        }
        container state {
```

```
          config false;
          description
            "State information relating to the BGP confederations";
          uses bgp-confederation_config;
        }
      }

      uses bgp-mp:bgp-use-multiple-paths;

      uses bgp-graceful-restart;

      container afi-safis {
        description
          "Address family specific configuration";
        uses bgp-mp:bgp-common-afi-safi-list;
      }
    }

    grouping bgp-neighbors {
      description
        "BGP neighbors configured on the local system";
      list neighbor {
        key "neighbor-address";
        description
          "List of BGP neighbors configured on the local system,
          uniquely identified by peer IPv[46] address";

        leaf neighbor-address {
          type leafref {
            path "../config/neighbor-address";
          }
          description
            "Reference to the address of the BGP neighbor used as
            a key in the neighbor list";
        }

        uses bgp-neighbor-group;
      }
    }

    grouping bgp-peer-group {
      description
        "BGP peer-groups configured on the local system";
      list peer-group {
        key "peer-group-name";
        description
          "List of BGP peer-groups configured on the local system -
          uniquely identified by peer-group name";
```

```
        leaf peer-group-name {
          type leafref {
            path "../config/peer-group-name";
          }
          description
            "Reference to the name of the BGP peer-group used as a
            key in the peer-group list";
          }

          uses bgp-neighbor-group;
        }
      }

      grouping bgp-neighbor-group {
        description
          "Parameters related to a BGP neighbor or group";

        container config {
          description
            "Configuration parameters relating to the BGP neighbor or
            group";
          uses bgp-neighbor_config;
        }
        container state {
          config false;
          description
            "State information relating to the BGP neighbor or group";
          uses bgp-neighbor_config;
        }

        container timers {
          description
            "Timers related to a BGP neighbor or group";
          container config {
            description
              "Configuration parameters relating to timers used for the
              BGP neighbor or group";
            uses bgp-neighbor-timers_config;
          }
          container state {
            config false;
            description
              "State information relating to the timers used for the BGP
              neighbor or group";
            uses bgp-neighbor-timers_config;
          }
        }
```

```
      container transport {
        description
          "Transport session parameters for the BGP neighbor or group";
        container config {
          description
            "Configuration parameters relating to the transport
            session(s) used for the BGP neighbor or group";
          uses bgp-neighbor-transport_config;
        }
        container state {
          config false;
          description
            "State information relating to the transport session(s)
            used for the BGP neighbor or group";
          uses bgp-neighbor-transport_config;
        }
      }

      container error-handling {
        description
          "Error handling parameters used for the BGP neighbor or
          group";
        container config {
          description
            "Configuration parameters enabling or modifying the
            behavior or enhanced error handling mechanisms for the BGP
            neighbor or group";
          uses bgp-neighbor-error-handling_config;
        }
        container state {
          config false;
          description
            "State information relating to enhanced error handling
            mechanisms for the BGP neighbor or group";
          uses bgp-neighbor-error-handling_config;
        }
      }

      container logging-options {
        description
          "Logging options for events related to the BGP neighbor or
          group";
        container config {
          description
            "Configuration parameters enabling or modifying logging
            for events relating to the BGP neighbor or group";
          uses bgp-neighbor-logging-options_config;
        }
```

```
        container state {
          config false;
          description
            "State information relating to logging for the BGP neighbor
            or group";
          uses bgp-neighbor-logging-options_config;
        }
      }

      container ebgp-multihop {
        description
          "eBGP multi-hop parameters for the BGP neighbor or group";
        container config {
          description
            "Configuration parameters relating to eBGP multihop for the
            BGP neighbor or group";
          uses bgp-neighbor-multihop_config;
        }
        container state {
          config false;
          description
            "State information for eBGP multihop, for the BGP neighbor
            or group";
          uses bgp-neighbor-multihop_config;
        }
      }

      container route-reflector {
        description
          "Route reflector parameters for the BGP neighbor or group";
        container config {
          description
            "Configuraton parameters relating to route reflection
            for the BGP neighbor or group";
          uses bgp-neighbor-route-reflector_config;
        }
        container state {
          config false;
          description
            "State information relating to route reflection for the
            BGP neighbor or group";
          uses bgp-neighbor-route-reflector_config;
        }
      }

      container as-path-options {
        description
          "AS_PATH manipulation parameters for the BGP neighbor or
```

```
          group";
        container config {
          description
            "Configuration parameters relating to AS_PATH manipulation
            for the BGP peer or group";
          uses bgp-neighbor-as-path-options_config;
        }
        container state {
          config false;
          description
            "State information relating to the AS_PATH manipulation
            mechanisms for the BGP peer or group";
          uses bgp-neighbor-as-path-options_config;
        }
      }

      container add-paths {
        description
          "Parameters relating to the advertisement and receipt of
          multiple paths for a single NLRI (add-paths)";
        container config {
          description
            "Configuration parameters relating to ADD_PATHS";
          uses bgp-neighbor-add-paths_config;
        }
        container state {
          config false;
          description
            "State information associated with ADD_PATHS";
          uses bgp-neighbor-add-paths_config;
        }
      }

      container afi-safis {
        description
          "Per-address-family configuration parameters associated with
          the neighbor or group";
        uses bgp-mp:bgp-common-afi-safi-list;
      }

      uses bgp-graceful-restart;

      uses rpol:apply-policy-group;
    }

    grouping bgp-neighbor-neighbor-address_config {
      description
        "Configuration options relating to the BGP neighbor address";
```

```
      leaf neighbor-address {
          type inet:ip-address;
          description
            "Address of the BGP peer, either in IPv4 or IPv6";
      }
    }

    grouping bgp-peer-group-peer-group-name_config {
      description
        "Configuration options relating to the BGP peer-group name";

      leaf peer-group-name {
        type string;
        description
          "Name of the BGP peer-group";
      }
    }

    // add peer-group pointer only for the neighbor list
    augment /bgp/neighbors/neighbor/config {
      description
        "Augmentation to allow association of a neighbor with a
        peer-group";
      uses bgp-neighbor-peer-group_config;
    }

    augment /bgp/neighbors/neighbor/state {
      description
        "Augmentation to reflect the association of a neighbor with a
        peer-group";
      uses bgp-neighbor-peer-group_config;
    }

    augment /bgp/peer-groups/peer-group {
      description
        "Augmentation to add multipath configuration to a peer-group";
      uses bgp-mp:bgp-use-multiple-paths;
    }

    augment /bgp/neighbors/neighbor {
      description
        "Augmentation to add the multipath configuration to a
        neighbor";
      uses bgp-mp:bgp-use-multiple-paths-neighbor;
    }

    augment /bgp/peer-groups/peer-group/afi-safis/afi-safi {
      description
```

```
        "Augmentation to add multipath configuration to a peer-group
        on a per-AFI-SAFI basis";
      uses bgp-mp:bgp-use-multiple-paths;
    }

    augment /bgp/neighbors/neighbor/afi-safis/afi-safi {
      description
        "Augmentation to add multipath configuration to a neighbor
        on a per-AFI-SAFI basis";
      uses bgp-mp:bgp-use-multiple-paths-neighbor;
    }

    augment /bgp/global/afi-safis/afi-safi {
      description
        "Augmentation to add global instance specific AFI-SAFI
        configuration information";
      uses bgp-mp:bgp-global-afi-safi;
      uses bgp-mp:bgp-use-multiple-paths;
    }

    augment /bgp/peer-groups/peer-group/afi-safis/afi-safi {
      description
        "Augmentation that adds peer-group instance specific
        AFI-SAFI configuration information";
      uses bgp-mp:bgp-group-afi-safi;
    }

    augment /bgp/neighbors/neighbor/config {
      description
        "Augmentation adding the neighbor address to the
        neighbor configuration container";
      uses bgp-neighbor-neighbor-address_config;
    }

    augment /bgp/neighbors/neighbor/state {
      description
        "Augmentation adding the neighbor address to the
        neighbor state container";
      uses bgp-neighbor-neighbor-address_config;
    }

    augment /bgp/peer-groups/peer-group/config {
      description
        "Augmentation adding the peer-group name to the
        peer-group configuration container";
      uses bgp-peer-group-peer-group-name_config;
    }
```

```
     augment /bgp/peer-groups/peer-group/state {
       description
         "Augmentation adding the peer-group name to the
         peer-group state container";
       uses bgp-peer-group-peer-group-name_config;
     }

     // ************************************************************
     // *              Augmentations to add state                  *
     // *  (rjs: cleaner to have these in the base module to avoid *
     //     needing to specify which module - e.g. augment of      *
     //     /bgp:bgp/bgp:neighbors/bgp:neighbor...)                *
     // ************************************************************
     augment /bgp/neighbors/neighbor/state {
       description
         "Augmentation to add operational state related to a particular
         BGP neighbor";
       uses bgp-op:bgp-neighbor_state;
     }

     augment /bgp/neighbors/bgp:neighbor/state {
       description
         "Augmentation to add operational state related to a particular
         BGP neighbor";

       container messages {
         description
           "Counters for BGP messages sent and received from the
           neighbor";
         container sent {
           description
             "Counters relating to BGP messages sent to the neighbor";
           uses bgp-op:bgp-neighbor-message-counters-sent_state;
           }

         container received {
           description
             "Counters for BGP messages received from the neighbor";
           uses bgp-op:bgp-neighbor-message-counters-received_state;
         }
       }

       container queues {
         description
           "Counters related to queued messages associated with the
           BGP neighbor";
         uses bgp-op:bgp-neighbor-queue-counters_state;
       }
```

```
     }

     augment /bgp:bgp/bgp:neighbors/neighbor/timers/state {
       description
         "Augmentation to add the operational state of timers associated
         with the BGP neighbor";
       uses bgp-op:bgp-neighbor-timers_state;
     }

     augment /bgp/neighbors/neighbor/transport/state {
       description
         "Augmentation to add the operational state of the transport
         session associated with the BGP neighbor";
       uses bgp-op:bgp-neighbor-transport_state;
     }

     augment /bgp/neighbors/neighbor/error-handling/state {
       description
         "Augmentation to add the operational state of the error
         handling associated with the BGP neighbor";
       uses bgp-op:bgp-neighbor-error-handling_state;
     }

     augment /bgp/neighbors/neighbor/graceful-restart/state {
       description
         "Augmentation to add the operational state of graceful-restart
         associated with a BGP neighbor";
       uses bgp-op:bgp-afi-safi-graceful-restart_state;
     }

     augment /bgp/peer-groups/peer-group/state {
       description
         "Augmentation to add the operational state and counters
         relating to a BGP peer-group";
       uses bgp-op:bgp-peer-group_state;
     }

     augment /bgp/global/afi-safis/afi-safi/state {
       description
         "Augmentation to add operational state and counters
         on a per-AFI-SAFI basis to the global BGP router";
       uses bgp-op:bgp-global-afi-safi_state;
     }

     augment /bgp/neighbors/neighbor/afi-safis/afi-safi/state {
       description
         "Augmentation to add per-AFI-SAFI operational state
         and counters to the BGP neighbor";
```

```
      uses bgp-op:bgp-neighbor-afi-safi_state;
    }

    augment "/bgp/neighbors/neighbor/afi-safis/afi-safi/" +
      "graceful-restart/state" {
      description
        "Augmentation to add per-AFI-SAFI operational state for BGP
        graceful-restart";
      uses bgp-op:bgp-neighbor-afi-safi-graceful-restart_state;
    }

    // ************************************************************
    // *              module structure containers                 *
    // ************************************************************

    container bgp {
      presence "Container for BGP protocol hierarchy";
      description
        "Top-level configuration and state for the BGP router";

      container global {
        description
          "Global configuration for the BGP router";
          uses bgp-global-base;
          uses rpol:apply-policy-group;
      }

      container neighbors {
        description
          "Configuration for BGP neighbors";
        uses bgp-neighbors;
      }

      container peer-groups {
        description
          "Configuration for BGP peer-groups";
        uses bgp-peer-group;
      }
    }
  }
  <CODE ENDS>
```

5.2. BGP base types

```
  <CODE BEGINS> file bgp-types.yang
  module bgp-types {
      yang-version "1";
```

```
      namespace "http://openconfig.net/yang/bgp-types";

      prefix "bgp-types";

      import ietf-inet-types { prefix inet; }

      // meta
      organization
      "OpenConfig working group";

    contact
      "OpenConfig working group
      netopenconfig@googlegroups.com";

    description
      "This module contains general data definitions for use in BGP
      policy. It can be imported by modules that make use of BGP
      attributes";

    revision "2015-05-15" {
      description "Initial revision";
      reference "TBD";
    }


    identity bgp-capability {
      description "Base identity for a BGP capability";
    }

    identity MPBGP {
      base "bgp-capability";
      description
        "Multi-protocol extensions to BGP";
      reference "RFC2858";
    }

    identity ROUTE-REFRESH {
      base "bgp-capability";
      description
        "The BGP route-refresh functionality";
      reference "RFC2918";
    }

    identity ASN32 {
      base "bgp-capability";
      description
        "4-byte (32-bit) AS number functionality";
      reference "RFC6793";
```

```
     }

     identity GRACEFUL-RESTART {
       base "bgp-capability";
       description
         "Graceful restart functionality";
       reference "RFC4724";
     }

     identity ADD-PATHS {
       base "bgp-capability";
       description
         "BGP add-paths";
       reference "draft-ietf-idr-add-paths";
     }

     identity afi-safi-type {
       description
         "Base identity type for AFI,SAFI tuples for BGP-4";
       reference "RFC4760 - multiprotocol extensions for BGP-4";
     }

     identity ipv4-unicast {
       base afi-safi-type;
       description
         "IPv4 unicast (AFI,SAFI = 1,1)";
       reference "RFC4760";
     }

     identity ipv6-unicast {
       base afi-safi-type;
       description
         "IPv6 unicast (AFI,SAFI = 2,1)";
       reference "RFC4760";
     }

     identity ipv4-labelled-unicast {
       base afi-safi-type;
       description
         "Labelled IPv4 unicast (AFI,SAFI = 1,4)";
       reference "RFC3107";
     }

     identity ipv6-labelled-unicast {
       base afi-safi-type;
       description
         "Labelled IPv6 unicast (AFI,SAFI = 2,4)";
       reference "RFC3107";
```

```
     }

     identity l3vpn-ipv4-unicast {
       base afi-safi-type;
       description
         "Unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128)";
       reference "RFC4364";
     }

     identity l3vpn-ipv6-unicast {
       base afi-safi-type;
       description
         "Unicast IPv6 MPLS L3VPN (AFI,SAFI = 2,128)";
       reference "RFC4659";
     }

     identity l3vpn-ipv4-multicast {
       base afi-safi-type;
       description
         "Multicast IPv4 MPLS L3VPN (AFI,SAFI = 1,129)";
       reference "RFC6514";
     }

     identity l3vpn-ipv6-multicast {
       base afi-safi-type;
       description
         "Multicast IPv6 MPLS L3VPN (AFI,SAFI = 2,129)";
       reference "RFC6514";
     }

     identity l2vpn-vpls {
       base afi-safi-type;
       description
         "BGP-signalled VPLS (AFI,SAFI = 25,65)";
       reference "RFC4761";
     }

     identity l2vpn-evpn {
       base afi-safi-type;
       description
         "BGP MPLS Based Ethernet VPN (AFI,SAFI = 25,70)";
     }

     identity bgp-well-known-std-community {
       description
         "Reserved communities within the standard community space
         defined by RFC1997. These communities must fall within the
         range 0x00000000 to 0xFFFFFFFF";
```

```
      reference "RFC1997";
    }

    identity NO_EXPORT {
      base bgp-well-known-std-community;
      description
        "Do not export NLRI received carrying this community outside
        the bounds of this autonomous system, or this confederation if
        the local autonomous system is a confederation member AS. This
        community has a value of 0xFFFFFF01.";
      reference "RFC1997";
    }

    identity NO_ADVERTISE {
      base bgp-well-known-std-community;
      description
        "All NLRI received carrying this community must not be
        advertised to other BGP peers. This community has a value of
        0xFFFFFF02.";
      reference "RFC1997";
    }

    identity NO_EXPORT_SUBCONFED {
      base bgp-well-known-std-community;
      description
        "All NLRI received carrying this community must not be
        advertised to external BGP peers - including over confederation
        sub-AS boundaries. This community has a value of 0xFFFFFF03.";
      reference "RFC1997";
    }

    identity NOPEER {
      base bgp-well-known-std-community;
      description
        "An autonomous system receiving NLRI tagged with this community
        is advised not to readvertise the NLRI to external bi-lateral
        peer autonomous systems. An AS may also filter received NLRI
        from bilateral peer sessions when they are tagged with this
        community value";
      reference "RFC3765";
    }

    identity INTERNET {
      base bgp-well-known-std-community;
      description
        "A community used by some implementations with the value 0:0
        which represents all possible community values.";
    }
```

```
typedef bgp-session-direction {
  type enumeration {
    enum INBOUND {
      description
        "Refers to all NLRI received from the BGP peer";
    }
    enum OUTBOUND {
      description
        "Refers to all NLRI advertised to the BGP peer";
    }
  }
  description
    "Type to describe the direction of NLRI transmission";
}

typedef bgp-well-known-community-type {
  type identityref {
    base bgp-well-known-std-community;
  }
  description
    "Type definition for well-known IETF community attribute
    values";
  reference
    "IANA Border Gateway Protocol (BGP) Well Known Communities";
}


typedef bgp-std-community-type {
  // TODO: further refine restrictions and allowed patterns
  // 4-octet value:
  //  <as number> 2 octets
  //  <community value> 2 octets
  type union {
    type uint32 {
    // per RFC 1997, 0x00000000 - 0x0000FFFF and 0xFFFF0000 -
    // 0xFFFFFFFF are reserved
      range "65536..4294901759"; // 0x00010000..0xFFFEFFFF
    }
    type string {
      pattern '([0-9]+:[0-9]+)';
    }
  }
  description
    "Type definition for standard commmunity attributes";
  reference "RFC 1997 - BGP Communities Attribute";
}

typedef bgp-ext-community-type {
```

```
      // TODO: needs more work to make this more precise given the
      // variability of extended community attribute specifications
      // 8-octet value:
      //  <type> 2 octects
      //  <value> 6 octets
      type string {
        pattern '([0-9\.]+(:[0-9]+)?:[0-9]+)';
      }
      description
        "Type definition for extended community attributes";
      reference "RFC 4360 - BGP Extended Communities Attribute";
    }

    typedef bgp-community-regexp-type {
      // TODO: needs more work to decide what format these regexps can
      // take.
      type string;
      description
        "Type definition for communities specified as regular
        expression patterns";
    }

    typedef bgp-origin-attr-type {
      type enumeration {
        enum IGP {
          description "Origin of the NLRI is internal";
        }
        enum EGP {
          description "Origin of the NLRI is EGP";
        }
        enum INCOMPLETE {
          description "Origin of the NLRI is neither IGP or EGP";
        }
      }
      description
        "Type definition for standard BGP origin attribute";
      reference "RFC 4271 - A Border Gateway Protocol 4 (BGP-4),
        Sec 4.3";
    }

    typedef peer-type {
      type enumeration {
        enum INTERNAL {
          description "internal (iBGP) peer";
        }
        enum EXTERNAL {
          description "external (eBGP) peer";
        }
```

```
      }
      description
        "labels a peer or peer group as explicitly internal or
        external";
    }


    typedef remove-private-as-option {
      type enumeration {
        enum ALL {
          description "remove all private ASes in the path";
        }
        enum REPLACE {
          description "replace private ASes with local AS";
        }
      }
      description
        "set of options for configuring how private AS path numbers
        are removed from advertisements";
    }

    typedef percentage {
      type uint8 {
        range "0..100";
      }
      description
        "Integer indicating a percentage value";
    }

    typedef rr-cluster-id-type {
      type union {
        type uint32;
        type inet:ipv4-address;
      }
      description
        "union type for route reflector cluster ids:
        option 1: 4-byte number
        option 2: IP address";
    }

    typedef community-type {
      type enumeration {
        enum STANDARD {
          description "send only standard communities";
        }
        enum EXTENDED {
          description "send only extended communities";
        }
```

```
          enum BOTH {
            description "send both standard and extended communities";
          }
          enum NONE {
            description "do not send any community attribute";
          }
        }
        description
          "type describing variations of community attributes:
          STANDARD: standard BGP community [rfc1997]
          EXTENDED: extended BGP community [rfc4360]
          BOTH: both standard and extended community";
      }


    }
    <CODE ENDS>
```

5.3. BGP policy items

```
    <CODE BEGINS> file bgp-policy.yang
    module bgp-policy {

      yang-version "1";

      // namespace
      namespace "http://openconfig.net/yang/bgp-policy";

      prefix "bgp-pol";

      // import some basic types
      import ietf-inet-types { prefix inet; }
      import routing-policy {prefix rpol; }
      import policy-types { prefix pt; }
      import bgp-types { prefix bgp-types; }



      // meta
      organization
        "OpenConfig working group";

      contact
        "OpenConfig working group
        netopenconfig@googlegroups.com";

      description
```

```
       "This module contains data definitions for BGP routing policy.
       It augments the base routing-policy module with BGP-specific
       options for conditions and actions.";

     revision "2015-05-15" {
       description
         "Updated model to augment base routing-policy module";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     typedef bgp-as-path-prepend-repeat {
       type uint8;
       description
         "Option for the BGP as-prepend policy action.  Prepends the
         local AS number repeated n times";
     }

     typedef bgp-set-community-option-type {
       type enumeration {
         enum ADD {
           description
             "add the specified communities to the existing
             community attribute";
         }
         enum REMOVE {
           description
             "remove the specified communities from the
             existing community attribute";
         }
         enum REPLACE {
           description
             "replace the existing community attribute with
             the specified communities. If an empty set is
             specified, this removes the community attribute
             from the route.";
         }
       }
       description
         "Type definition for options when setting the community
         attribute in a policy action";
```

```
     }

     typedef bgp-next-hop-type {
       type union {
         type inet:ip-address;
         type enumeration {
           enum SELF {
             description "special designation for local router's own
             address, i.e., next-hop-self";
           }
         }
       }
       description
         "type definition for specifying next-hop in policy actions";
     }

     typedef bgp-set-med-type {
       type union {
         type uint32;
         type string {
           pattern "^[+-][0-9]+";
         }
         type enumeration {
           enum IGP {
             description "set the MED value to the IGP cost toward the
             next hop for the route";
           }
         }
       }
       description
         "Type definition for specifying how the BGP MED can
         be set in BGP policy actions. The three choices are to set
         the MED directly, increment/decrement using +/- notation,
         and setting it to the IGP cost (predefined value).";
     }

     // grouping statements

     grouping bgp-match-conditions {
       description
         "Condition statement definitions for checking membership in a
         defined set";

       container match-community-set {
         presence
           "The presence of this container indicates that the routes
           should match the referenced community-set";
```

```
        description
          "Match a referenced community-set according to the logic
          defined in the match-set-options leaf";

        leaf community-set {
          type leafref {
            path "/rpol:routing-policy/rpol:defined-sets/" +
              "bgp-pol:bgp-defined-sets/bgp-pol:community-sets/" +
              "bgp-pol:community-set/bgp-pol:community-set-name";
            require-instance true;
          }
          description
            "References a defined community set";
        }
        uses rpol:match-set-options-group;
      }

      container match-ext-community-set {
        presence
          "The presence of this container indicates that the routes
          should match the referenced extended community set";

        description
          "Match a referenced extended community-set according to the
          logic defined in the match-set-options leaf";

        leaf ext-community-set {
          type leafref {
            path "/rpol:routing-policy/rpol:defined-sets/" +
              "bgp-pol:bgp-defined-sets/bgp-pol:ext-community-sets/" +
              "bgp-pol:ext-community-set/" +
              "bgp-pol:ext-community-set-name";
            require-instance true;
          }
          description "References a defined extended community set";
        }
        uses rpol:match-set-options-group;
      }

      container match-as-path-set {
        presence
          "The presence of this container indicates that the route
          should match the referenced as-path set";

        description
          "Match a referenced as-path set according to the logic
          defined in the match-set-options leaf";
```

```
        leaf as-path-set {
          type leafref {
            path "/rpol:routing-policy/rpol:defined-sets/" +
              "bgp-pol:bgp-defined-sets/bgp-pol:as-path-sets/" +
              "bgp-pol:as-path-set/bgp-pol:as-path-set-name";
            require-instance true;
          }
          description "References a defined AS path set";
        }
        uses rpol:match-set-options-group;
      }
    }

    grouping bgp-attribute-conditions {
      description
        "Condition statement definitions for comparing a BGP route
        attribute to a specified value";

      leaf med-eq {
        type uint32;
        description
          "Condition to check if the received MED value is equal to
          the specified value";
      }

      leaf origin-eq {
        type bgp-types:bgp-origin-attr-type;
        description
          "Condition to check if the route origin is equal to the
          specified value";
      }

      leaf-list next-hop-in {
        type inet:ip-address;
        description
          "List of next hop addresses to check for in the route
          update";
      }

      leaf local-pref-eq {
        type uint32;
        // TODO: add support for other comparisons if needed
        description
          "Condition to check if the local pref attribute is equal to
          the specified value";
      }

      container community-count {
```

```
        presence "node is present in the config data to indicate a
        community-count condition";

        description
          "Value and comparison operations for conditions based on the
          number of communities in the route update";

        uses pt:attribute-compare-operators;

      }

      container as-path-length {

        presence "node is present in the config data to indicate a
        as-path-length condition";

        description
          "Value and comparison operations for conditions based on the
          length of the AS path in the route update";

        uses pt:attribute-compare-operators;
      }

      leaf route-type {
        // TODO: verify extent of vendor support for this comparison
        type enumeration {
          enum INTERNAL {
            description "route type is internal";
          }
          enum EXTERNAL {
            description "route type is external";
          }
        }
        description
          "Condition to check the route type in the route update";
      }
    }


    // augment statements

    augment "/rpol:routing-policy/rpol:defined-sets" {
      description "adds BGP defined sets container to routing policy
      model";

      container bgp-defined-sets {
        description
          "BGP-related set definitions for policy match conditions";
```

```
        container community-sets {
          description
            "Enclosing container for community sets";

          list community-set {
            key community-set-name;
            description
                "Definitions for community sets";

            leaf community-set-name {
              type string;
              mandatory true;
              description
                "name / label of the community set -- this is used to
                reference the set in match conditions";
            }

            leaf-list community-member {
              type union {
                type bgp-types:bgp-std-community-type;
                type bgp-types:bgp-community-regexp-type;
                type bgp-types:bgp-well-known-community-type;
              }
              description
                "members of the community set";
            }
          }
        }

        container ext-community-sets {
          description
            "Enclosing container for extended community sets";

          list ext-community-set {
            key ext-community-set-name;
            description
                "Definitions for extended community sets";

            leaf ext-community-set-name {
              type string;
              description
                "name / label of the extended community set -- this is
                used to reference the set in match conditions";
            }

            leaf-list ext-community-member {
              type union {
                type bgp-types:bgp-ext-community-type;
```

```
                // TODO: is regexp support needed for extended
                // communities?
                type bgp-types:bgp-community-regexp-type;
              }
              description
                  "members of the extended community set";
            }
          }
        }

        container as-path-sets {
          description
            "Enclosing container for AS path sets";

          list as-path-set {
            key as-path-set-name;
            description
                "Definitions for AS path sets";

            leaf as-path-set-name {
              type string;
              description
                "name of the AS path set -- this is used to reference
                the set in match conditions";
            }

            leaf-list as-path-set-member {
              // TODO: need to refine typedef for AS path expressions
              type string;
              description
                  "AS path expression -- list of ASes in the set";
            }
          }
        }
      }
    }

    augment "/rpol:routing-policy/rpol:policy-definitions/" +
      "rpol:policy-definition/rpol:statements/rpol:statement/" +
      "rpol:conditions" {
      description "BGP policy conditions added to routing policy
      module";

      container bgp-conditions {
        description "Policy conditions for matching
        BGP-specific defined sets or comparing BGP-specific
        attributes";
```

```
        uses bgp-match-conditions;
        uses bgp-attribute-conditions;
      }
    }

    augment "/rpol:routing-policy/rpol:policy-definitions/" +
      "rpol:policy-definition/rpol:statements/rpol:statement/" +
      "rpol:actions" {
      description "BGP policy actions added to routing policy
      module";

      container bgp-actions {
        description
          "Definitions for policy action statements that
          change BGP-specific attributes of the route";

        container set-as-path-prepend {

          presence "node is present in the config data to use the AS
        prepend action";
          description
              "action to prepend local AS number to the AS-path a
          specified number of times";

          leaf repeat-n {
            type uint8;
            description "number of times to prepend the local AS
            number";
          }
        }

        container set-community {
          presence "node is present in the config data when
          set-community action is used";
          description
            "action to set the community attributes of the route, along
            with options to modify how the community is modified";

          choice set-community-method {
            description
              "Option to set communities using an inline list or
              reference to an existing defined set.";

            case inline {
              leaf-list communities {
                type union {
                  type bgp-types:bgp-std-community-type;
                  type bgp-types:bgp-well-known-community-type;
```

```
                }
                description
                  "Set the community values for the update inline with
                  a list.";
              }
            }
            case reference {
              leaf community-set-ref {
                type leafref {
                  path "/rpol:routing-policy/rpol:defined-sets/" +
                    "bgp-pol:bgp-defined-sets/" +
                    "bgp-pol:community-sets/bgp-pol:community-set/" +
                    "bgp-pol:community-set-name";
                  require-instance true;
                }
                description
                  "References a defined community set by name";
              }
            }
          }
          leaf options {
            type bgp-set-community-option-type;
            description
              "Options for modifying the community attribute with
              the specified values.  These options apply to both
              methods of setting the community attribute.";
          }
        }

        container set-ext-community {

          presence "node is present in the config data when
          set-community action is used";
          description
            "Action to set the extended community attributes of the
            route, along with options to modify how the community is
            modified";

          choice set-ext-community-method {
            description
              "Option to set communities using an inline list or
              reference to an existing defined set.";

            case inline {
              leaf-list communities {
                type union {
                  type bgp-types:bgp-ext-community-type;
                  type bgp-types:bgp-well-known-community-type;
```

```
                }
                description
                  "Set the community values for the update inline with
                  a list.";
              }
            }
            case reference {
              leaf ext-community-set-ref {
                type leafref {
                  path "/rpol:routing-policy/rpol:defined-sets/" +
                    "bgp-pol:bgp-defined-sets/" +
                    "bgp-pol:ext-community-sets/" +
                    "bgp-pol:ext-community-set/" +
                    "bgp-pol:ext-community-set-name";
                  require-instance true;
                }
                description
                  "References a defined extended community set by
                  name";
              }
            }
          }
          leaf options {
            type bgp-set-community-option-type;
            description
              "options for modifying the extended community
              attribute with the specified values. These options
              apply to both methods of setting the community
              attribute.";
          }
        }

        leaf set-route-origin {
          type bgp-types:bgp-origin-attr-type;
          description "set the origin attribute to the specified
          value";
        }

        leaf set-local-pref {
          type uint32;
          description "set the local pref attribute on the route
          update";
        }

        leaf set-next-hop {
          type bgp-next-hop-type;
          description "set the next-hop attribute in the route update";
        }
```

```
        leaf set-med {
          type bgp-set-med-type;
          description "set the med metric attribute in the route
          update";
        }
      }
    }

    // rpc statements

    // notification statements
  }
  <CODE ENDS>
```

5.4. BGP multiprotocol items

```
  <CODE BEGINS> file bgp-multiprotocol.yang
  module bgp-multiprotocol {

    yang-version "1";

    // namespace
    namespace "http://openconfig.net/yang/bgp-multiprotocol";

    prefix "bgp-mp";

    // import some basic inet types
    import routing-policy { prefix rpol; }
    import bgp-types { prefix bgp-types; }
    import bgp-operational { prefix bgp-op; }

    // meta
    organization
      "OpenConfig working group";

    contact
      "OpenConfig working group
      netopenconfig@googlegroups.com";

    description
      "This module is part of a YANG model for BGP protocol
      configuration, focusing on configuration of multiprotocol
      BGP, in particular various relevant address families (AFI) and
      sub-address families (SAFI).

      Identities (rather than enumerated types) are used to identify
      each AFI / SAFI type to make it easier for users to extend to
```

```
       pre-standard or custom AFI/SAFI types.  This module is only
       intended to capture the most";


     revision "2015-05-15" {
       description
         "Refactored multiprotocol module";
       reference "TBD";
     }

     grouping ipv4-unicast-group {
       description
         "Group for IPv4 Unicast configuration options";

       container ipv4-unicast {
         when "../afi-safi-name = 'bgp-mp:ipv4-unicast'" {
           description
             "Include this container for IPv4 Unicast specific
             configuration";
         }

         description "IPv4 unicast configuration options";

         // include common IPv[46] unicast options
         uses ipv4-ipv6-unicast-common;

         // placeholder for IPv4 unicast  specific configuration
       }
     }

     grouping ipv6-unicast-group {
       description
         "Group for IPv6 Unicast configuration options";

       container ipv6-unicast {
         when "../afi-safi-name = 'bgp-mp:ipv6-unicast'" {
           description
             "Include this container for IPv6 Unicast specific
             configuration";
         }

         description "IPv6 unicast configuration options";

         // include common IPv[46] unicast options
         uses ipv4-ipv6-unicast-common;

         // placeholder for IPv6 unicast specific configuration
         // options
```

```
      }
    }

    grouping ipv4-labelled-unicast-group {
      description
        "Group for IPv4 Labelled Unicast configuration options";

      container ipv4-labelled-unicast {
        when "../afi-safi-name = 'bgp-mp:ipv4-labelled-unicast'" {
          description
            "Include this container for IPv4 Labelled Unicast specific
            configuration";
        }

        description "IPv4 Labelled Unicast configuration options";

        uses all-afi-safi-common;

        // placeholder for IPv4 Labelled Unicast specific config
        // options
      }
    }

    grouping ipv6-labelled-unicast-group {
      description
        "Group for IPv6 Labelled Unicast configuration options";

      container ipv6-labelled-unicast {
        when "../afi-safi-name = 'bgp-mp:ipv6-labelled-unicast'" {
          description
            "Include this container for IPv6 Labelled Unicast specific
            configuration";
        }

        description "IPv6 Labelled Unicast configuration options";

        uses all-afi-safi-common;

        // placeholder for IPv6 Labelled Unicast specific config
        // options.
      }
    }

    grouping l3vpn-ipv4-unicast-group {
      description
        "Group for IPv4 Unicast L3VPN configuration options";

      container l3vpn-ipv4-unicast {
```

```
        when "../afi-safi-name = 'bgp-mp:l3vpn-ipv4-unicast'" {
          description
            "Include this container for IPv4 Unicast L3VPN specific
            configuration";
        }

        description "Unicast IPv4 L3VPN configuration options";

        // include common L3VPN configuration options
        uses l3vpn-ipv4-ipv6-unicast-common;

        // placeholder for IPv4 Unicast L3VPN specific config options.
      }
    }

    grouping l3vpn-ipv6-unicast-group {
      description
        "Group for IPv6 Unicast L3VPN configuration options";

      container l3vpn-ipv6-unicast {
        when "../afi-safi-name = 'bgp-mp:l3vpn-ipv6-unicast'" {
          description
            "Include this container for unicast IPv6 L3VPN specific
            configuration";
        }

        description "Unicast IPv6 L3VPN configuration options";

        // include common L3VPN configuration options
        uses l3vpn-ipv4-ipv6-unicast-common;

        // placeholder for IPv6 Unicast L3VPN specific configuration
        // options
      }
    }

    grouping l3vpn-ipv4-multicast-group {
      description
        "Group for IPv4 L3VPN multicast configuration options";

      container l3vpn-ipv4-multicast {
        when "../afi-safi-name = 'bgp-mp:l3vpn-ipv4-multicast'" {
          description
            "Include this container for multicast IPv6 L3VPN specific
            configuration";
        }

        description "Multicast IPv4 L3VPN configuration options";
```

```
        // include common L3VPN multicast options
        uses l3vpn-ipv4-ipv6-multicast-common;

        // placeholder for IPv4 Multicast L3VPN specific configuration
        // options
      }
    }

    grouping l3vpn-ipv6-multicast-group {
      description
        "Group for IPv6 L3VPN multicast configuration options";

      container l3vpn-ipv6-multicast {
        when "../afi-safi-name = 'bgp-mp:l3vpn-ipv6-multicast'" {
          description
            "Include this container for multicast IPv6 L3VPN specific
            configuration";
        }

        description "Multicast IPv6 L3VPN configuration options";

        // include common L3VPN multicast options
        uses l3vpn-ipv4-ipv6-multicast-common;

        // placeholder for IPv6 Multicast L3VPN specific configuration
        // options
      }
    }

    grouping l2vpn-vpls-group {
      description
        "Group for BGP-signalled VPLS configuration options";

      container l2vpn-vpls {
        when "../afi-safi-name = 'bgp-mp:l2vpn-vpls'" {
          description
            "Include this container for BGP-signalled VPLS specific
            configuration";
        }

        description "BGP-signalled VPLS configuration options";

        // include common L2VPN options
        uses l2vpn-common;

        // placeholder for BGP-signalled VPLS specific configuration
        // options
      }
```

```
     }

     grouping l2vpn-evpn-group {
       description
         "Group for BGP EVPN configuration options";

       container l2vpn-evpn {
         when "../afi-safi-name = 'bgp-mp:l2vpn-evpn'" {
           description
             "Include this container for BGP EVPN specific
             configuration";
         }

         description "BGP EVPN configuration options";

         // include common L2VPN options
         uses l2vpn-common;

         // placeholder for BGP EVPN specific configuration options
       }
     }



     grouping bgp-route-selection-options_config {
       description
         "Set of configuration options that govern best
          path selection.";

       leaf always-compare-med {
         type boolean;
         default "false";
         description
           "Compare multi-exit discriminator (MED) value from
           different ASes when selecting the best route.  The
           default behavior is to only compare MEDs for paths
           received from the same AS.";
       }

       leaf ignore-as-path-length {
         type boolean;
         default "false";
         description
           "Ignore the AS path length when selecting the best path.
           The default is to use the AS path length and prefer paths
           with shorter length.";
       }
```

```
        leaf external-compare-router-id {
          type boolean;
          default "true";
          description
            "When comparing similar routes received from external
            BGP peers, use the router-id as a criterion to select
            the active path.";
        }

        leaf advertise-inactive-routes {
          type boolean;
          default "false";
          description
            "Advertise inactive routes to external peers.  The
            default is to only advertise active routes.";
        }

        leaf enable-aigp {
          type boolean;
          default false;
          description
            "Flag to enable sending / receiving accumulated IGP
            attribute in routing updates";
        }

        leaf ignore-next-hop-igp-metric {
          type boolean;
          default "false";
          description
            "Ignore the IGP metric to the next-hop when calculating
            BGP best-path. The default is to select the route for
            which the metric to the next-hop is lowest";
        }
      }

      grouping bgp-use-multiple-paths-ebgp-as-options_config {
        description
          "Configuration parameters specific to eBGP multipath applicable
          to all contexts";

        leaf allow-multiple-as {
         type boolean;
         default "false";
         description
          "Allow multipath to use paths from different neighbouring
          ASes.  The default is to only consider multiple paths from
          the same neighbouring AS.";
        }
```

```
     }

     grouping bgp-use-multiple-paths-ebgp_config {
       description
         "Configuration parameters relating to multipath for eBGP";

       uses bgp-use-multiple-paths-ebgp-as-options_config;

       leaf maximum-paths {
        type uint32;
        default 1;
        description
         "Maximum number of parallel paths to consider when using
         BGP multipath. The default is use a single path.";
       }
     }

     grouping bgp-use-multiple-paths-ibgp_config {
       description
         "Configuration parmaeters relating to multipath for iBGP";

       leaf maximum-paths {
         type uint32;
         default 1;
         description
           "Maximum number of parallel paths to consider when using
           iBGP multipath. The default is to use a single path";
       }
     }

     grouping bgp-use-multiple-paths {
       description
         "Configuration parameters relating to multipath for BGP - both
         iBGP and eBGP";

       container use-multiple-paths {
         description
           "Parameters related to the use of multiple paths for the
           same NLRI";

         container config {
           description
             "Configuration parameters relating to multipath";
           uses bgp-use-multiple-paths_config;
         }
         container state {
           description
             "State parameters relating to multipath";
```

```
          uses bgp-use-multiple-paths_config;
        }

        container ebgp {
          description
            "Multipath parameters for eBGP";
          container config {
            description
              "Configuration parameters relating to eBGP multipath";
            uses bgp-use-multiple-paths-ebgp_config;
          }
          container state {
            config false;
            description
              "State information relating to eBGP multipath";
            uses bgp-use-multiple-paths-ebgp_config;
          }
        }

        container ibgp {
          description
            "Multipath parameters for iBGP";
          container config {
            description
              "Configuration parameters relating to iBGP multipath";
            uses bgp-use-multiple-paths-ibgp_config;
          }
          container state {
            config false;
            description
              "State information relating to iBGP multipath";
            uses bgp-use-multiple-paths-ibgp_config;
          }
        }
      }
    }

    grouping bgp-use-multiple-paths-neighbor {
      description
        "Per-neighbor configuration for multipath for BGP";

      container use-multiple-paths {
        description
          "Parameters related to the use of multiple-paths for the same
          NLRI when they are received only from this neighbor";

        container config {
          description
```

```
            "Configuration parameters relating to multipath";
          uses bgp-use-multiple-paths_config;
        }
        container state {
          description
            "State parameters relating to multipath";
          uses bgp-use-multiple-paths_config;
        }

        container ebgp {
          description
            "Multipath configuration for eBGP";
          container config {
            description
              "Configuration parameters relating to eBGP multipath";
            uses bgp-use-multiple-paths-ebgp-as-options_config;
          }
          container state {
            config false;
            description
              "State information relating to eBGP multipath";
            uses bgp-use-multiple-paths-ebgp-as-options_config;
          }
        }
      }
    }

    grouping bgp-use-multiple-paths_config {
      description
        "Generic configuration options relating to use of multiple
        paths for a referenced AFI-SAFI, group or neighbor";

      leaf enabled {
        type boolean;
        default false;
        description
          "Whether the use of multiple paths for the same NLRI is
          enabled for the neighbor. This value is overridden by
          any more specific configuration value.";
      }
    }

    grouping bgp-afi-safi-graceful-restart_config {
      description
        "BGP graceful restart parameters that apply on a per-AFI-SAFI
        basis";

      leaf enabled {
```

```
        type boolean;
        default false;
        description
          "This leaf indicates whether graceful-restart is enabled for
          this AFI-SAFI";
      }
    }

    grouping bgp-afi-safi_config {
      description
        "Configuration parameters used for all BGP AFI-SAFIs";

      leaf afi-safi-name {
        type identityref {
          base bgp-types:afi-safi-type;
        }
        description "AFI,SAFI";
      }

      leaf enabled {
        type boolean;
        default false;
        description
          "This leaf indicates whether the IPv4 Unicast AFI,SAFI is
          enabled for the neighbour or group";
      }
    }

    grouping all-afi-safi-common-prefix-limit_config {
      description
        "Configuration parameters relating to prefix-limits for an
        AFI-SAFI";

      leaf max-prefixes {
        type uint32;
        description
          "Maximum number of prefixes that will be accepted
          from the neighbour";
      }

      leaf shutdown-threshold-pct {
        type bgp-types:percentage;
        description
          "Threshold on number of prefixes that can be received
          from a neighbour before generation of warning messages
          or log entries. Expressed as a percentage of
          max-prefixes";
      }
```

```
      leaf restart-timer {
        type decimal64 {
          fraction-digits 2;
        }
        units "seconds";
        description
          "Time interval in seconds after which the BGP session
          is re-established after being torn down due to exceeding
          the max-prefix limit.";
      }
    }

    grouping ipv4-ipv6-unicast-common_config {
      description
        "Common configuration parameters for IPv4 and IPv6 Unicast
        address families";

      leaf send-default-route {
        type boolean;
        default "false";
        description
          "If set to true, send the default-route to the neighbour(s)";
      }
    }

    grouping all-afi-safi-common {
      description
        "Grouping for configuration common to all AFI,SAFI";

      container prefix-limit {
        description
          "Configure the maximum number of prefixes that will be
          accepted from a peer";

        container config {
          description
            "Configuration parameters relating to the prefix
            limit for the AFI-SAFI";
          uses all-afi-safi-common-prefix-limit_config;
        }
        container state {
          config false;
          description
            "State information relating to the prefix-limit for the
            AFI-SAFI";
          uses all-afi-safi-common-prefix-limit_config;
        }
      }
```

```
     }

     grouping ipv4-ipv6-unicast-common {
       description
         "Common configuration that is applicable for IPv4 and IPv6
         unicast";

       // include common afi-safi options.
       uses all-afi-safi-common;

       // configuration options that are specific to IPv[46] unicast
       container config {
         description
           "Configuration parameters for common IPv4 and IPv6 unicast
           AFI-SAFI options";
         uses ipv4-ipv6-unicast-common_config;
       }
       container state {
         config false;
         description
           "State information for common IPv4 and IPv6 unicast
           parameters";
         uses ipv4-ipv6-unicast-common_config;
       }
     }

     grouping l3vpn-ipv4-ipv6-unicast-common {
       description
         "Common configuration applied across L3VPN for IPv4
          and IPv6";

       // placeholder -- specific configuration options that are generic
       // across IPv[46] unicast address families.
       uses all-afi-safi-common;
     }

     grouping l3vpn-ipv4-ipv6-multicast-common {
       description
         "Common configuration applied across L3VPN for IPv4
         and IPv6";

       // placeholder -- specific configuration options that are
       // generic across IPv[46] multicast address families.
       uses all-afi-safi-common;
     }

     grouping l2vpn-common {
       description
```

```
        "Common configuration applied across L2VPN address
        families";

      // placeholder -- specific configuration options that are
      // generic across L2VPN address families
      uses all-afi-safi-common;
    }

    grouping bgp-route-selection-options {
      description
        "Parameters relating to the BGP route selection process";

      container route-selection-options {
        description
          "Parameters relating to options for route selection";
        container config {
          description
            "Configuration parameters relating to route selection
            options";
          uses bgp-route-selection-options_config;
        }
        container state {
          config false;
          description
            "State information for the route selection options";
          uses bgp-route-selection-options_config;
        }
      }
    }

    // *********** STRUCTURE GROUPINGS **********************

    grouping bgp-global-afi-safi {
      description
        "Parameters and route selection options for MP-BGP
        specific to the Global AFI-SAFI";
      uses bgp-route-selection-options;
    }

    grouping bgp-group-afi-safi {
      description
        "Parameters and route selection options for MP-BGP
        specific to peer groups";
      uses bgp-route-selection-options;
    }

    grouping bgp-common-afi-safi-list {
      description
```

```
        "List of address-families associated with the BGP instance,
        a peer-group or neighbor";

      list afi-safi {
        key "afi-safi-name";

        description
          "AFI,SAFI configuration available for the
          neighbour or group";


        leaf afi-safi-name {
          type leafref {
            path "../config/afi-safi-name";
          }
          description
            "Reference to the AFI-SAFI name used as a key
            for the AFI-SAFI list";
        }

        container graceful-restart {
          description
            "Parameters relating to BGP graceful-restart";
          container config {
            description
              "Configuration options for BGP graceful-restart";
            uses bgp-afi-safi-graceful-restart_config;
          }
          container state {
            config false;
            description
              "State information for BGP graceful-restart";
            uses bgp-afi-safi-graceful-restart_config;
          }
        }

        container config {
          description
            "Configuration parameters for the AFI-SAFI";
          uses bgp-afi-safi_config;
        }
        container state {
          config false;
          description
            "State information relating to the AFI-SAFI";
          uses bgp-afi-safi_config;
          uses bgp-op:bgp-afi-safi_state;
        }
```

```
        // import and export policy included for the afi/safi
        uses rpol:apply-policy-group;

        uses ipv4-unicast-group;
        uses ipv6-unicast-group;
        uses ipv4-labelled-unicast-group;
        uses ipv6-labelled-unicast-group;
        uses l3vpn-ipv4-unicast-group;
        uses l3vpn-ipv6-unicast-group;
        uses l3vpn-ipv4-multicast-group;
        uses l3vpn-ipv6-multicast-group;
        uses l2vpn-vpls-group;
        uses l2vpn-evpn-group;
      }
    }
  }
  <CODE ENDS>
```

5.5. BGP operational data items

```
  <CODE BEGINS> file bgp-operational.yang
  module bgp-operational {

    yang-version "1";

    // namespace
    // TODO: change to an ietf or other more generic namespace
    namespace "http://openconfig.net/yang/bgp-operational";

    prefix "bgp-op";

    // import some basic inet types
    import ietf-inet-types { prefix inet; }
    import ietf-yang-types { prefix yang; }
    import bgp-types { prefix bgp-types; }

    // meta

    organization
      "OpenConfig working group";

    contact
      "OpenConfig working group
      netopenconfig@googlegroups.com";

    description
      "This module is part of a YANG model for BGP protocol
```

```
      configuration, focusing on operational data (i.e., state
      variables) related to BGP operations";


    revision "2015-05-15" {
      description
        "Initial revision";
      reference "TBD";
    }

    // extension statements

    // feature statements

    // identity statements

    // typedef statements

    // grouping statements

    grouping bgp-counters-message-types_common {
      description
        "Grouping of BGP message types, included for re-use
        across counters";

      leaf UPDATE {
        type uint64;
        description
          "Number of BGP UPDATE messages announcing, withdrawing
          or modifying paths exchanged.";
      }

      leaf NOTIFICATION {
        type uint64;
        description
          "Number of BGP NOTIFICATION messages indicating an
          error condition has occurred exchanged.";
      }
    }


    grouping bgp-context-pfx-path-counters_common {
      description
        "Grouping containing common counters relating to prefixes and
        paths";

      leaf total-paths {
        type uint32;
```

```
        description
          "Total number of BGP paths within the context";
      }

      leaf total-prefixes {
        type uint32;
        description
          "";
      }
    }

    grouping bgp-global_state {
      description
        "Grouping containing operational parameters relating to the
        global BGP instance";
      uses bgp-context-pfx-path-counters_common;
    }

    grouping bgp-global-afi-safi_state {
      description
        "Grouping containing operational parameters relating to each
        AFI-SAFI within the BGP global instance";
      uses bgp-context-pfx-path-counters_common;
    }

    grouping bgp-peer-group_state {
      description
        "Grouping containing operational parameters relating to a BGP
        peer group";
      uses bgp-context-pfx-path-counters_common;
    }

    grouping bgp-neighbor_state {
      description
        "Grouping containing operational state variables relating to a
        BGP neighbor";

      leaf session-state {
        type enumeration {
            enum IDLE {
              description
                "neighbor is down, and in the Idle state of the
                FSM";
            }
            enum CONNECT {
              description
                "neighbor is down, and the session is waiting for
                the underlying transport session to be established";
```

```
            }
            enum ACTIVE {
              description
                "neighbor is down, and the local system is awaiting
                a conncetion from the remote peer";
            }
            enum OPENSENT {
              description
                "neighbor is in the process of being established.
                The local system has sent an OPEN message";
            }
            enum OPENCONFIRM {
              description
                "neighbor is in the process of being established.
                The local system is awaiting a NOTIFICATION or
                KEEPALIVE message";
            }
            enum ESTABLISHED {
              description
                "neighbor is up - the BGP session with the peer is
                established";
            }
          }
        description
          "Operational state of the BGP peer";
      }

      leaf-list supported-capabilities {
        type identityref {
          base bgp-types:bgp-capability;
        }
        description
          "BGP capabilities negotiated as supported with the peer";
      }
    }

    grouping bgp-neighbor-afi-safi_state {
      description
        "Operational state on a per-AFI-SAFI basis for a BGP
        neighbor";

      leaf active {
        type boolean;
        description
          "This value indicates whether a particular AFI-SAFI has
          been succesfully negotiated with the peer. An AFI-SAFI
          may be enabled in the current running configuration, but a
          session restart may be required in order to negotiate the new
```

```
          capability.";
      }

      uses bgp-neighbor-prefix-counters_state;
    }

    grouping bgp-neighbor-prefix-counters_state {
      description
        "Counters for BGP neighbor sessions";

      container prefixes {
        description "Prefix counters for the BGP session";
        leaf received {
          type uint32;
          description
            "The number of prefixes received from the neighbor";
        }

        leaf sent {
          type uint32;
          description
            "The number of prefixes advertised to the neighbor";
        }

        leaf installed {
          type uint32;
          description
            "The number of advertised prefixes installed in the
            Loc-RIB";
        }
      }
    }

    grouping bgp-neighbor-message-counters-sent_state {
      description
        "Counters relating to messages sent to a BGP neighbor";
      uses bgp-counters-message-types_common;
    }

    grouping bgp-neighbor-message-counters-received_state {
      description
        "Counters relating to the mesages received from a BGP
        neighbor";
      uses bgp-counters-message-types_common;
    }

    grouping bgp-neighbor-queue-counters_state {
      description
```

```
        "Counters relating to the message queues associated with the
        BGP peer";
      leaf input {
        type uint32;
        description
          "The number of messages received from the peer currently
          queued";
      }

      leaf output {
        type uint32;
        description
          "The number of messages queued to be sent to the peer";
      }
    }

    grouping bgp-neighbor-transport_state {
      description
        "Operational state parameters relating to the transport session
        used for the BGP session";

      leaf local-port {
        type inet:port-number;
        description
          "Local TCP port being used for the TCP session supporting
          the BGP session";
      }

      leaf remote-address {
        type inet:ip-address;
        description
          "Remote port being used by the peer for the TCP session
          supporting the BGP session";
      }

      leaf remote-port {
        type inet:port-number;
        description
          "Remote address to which the BGP session has been
          established";
      }
    }

    grouping bgp-neighbor-error-handling_state {
      description
        "Operational state parameters relating to enhanced error
        error handling for BGP";
```

```
        leaf erroneous-update-messages {
          type uint32;
          description
            "The number of BGP UPDATE messages for which the
            treat-as-withdraw mechanism has been applied based
            on erroneous message contents";
        }
      }

      grouping bgp-neighbor-timers_state {
        description
          "Operational state parameters relating to BGP timers associated
          with the BGP session";

        leaf uptime {
          type yang:timeticks;
          description
            "This timer determines the amount of time since the
            BGP last transitioned in or out of the Established
            state";
        }

        leaf negotiated-hold-time {
          type decimal64 {
            fraction-digits 2;
          }
          description
            "The negotiated hold-time for the BGP session";
        }
      }

      grouping bgp-afi-safi_state {
        description
          "Operational state information relevant to all address
          families that may be carried by the BGP session";

        // placeholder - options in this container are
        // valid in both the global and per-neighbor
        // paths

      }

      grouping bgp-afi-safi-graceful-restart_state {
        description
          "Operational state information relevant to graceful restart
          for BGP";

        leaf peer-restart-time {
```

```
        type uint16 {
          range 0..4096;
        }
        description
          "The period of time (advertised by the peer) that
          the peer expects a restart of a BGP session to
          take";
      }

      leaf peer-restarting {
        type boolean;
        description
          "This flag indicates whether the remote neighbor is currently
          in the process of restarting, and hence received routes are
          currently stale";
      }

      leaf local-restarting {
        type boolean;
        description
          "This flag indicates whether the local neighbor is currently
          restarting. The flag is unset after all NLRI have been
          advertised to the peer, and the End-of-RIB (EOR) marker has
          been unset";
      }

      leaf mode {
        type enumeration {
          enum HELPER-ONLY {
            description
              "The local router is operating in helper-only mode, and
              hence will not retain forwarding state during a local
              session restart, but will do so during a restart of the
              remote peer";
          }
          enum BILATERAL {
            description
              "The local router is operating in both helper mode, and
              hence retains forwarding state during a remote restart,
              and also maintains forwarding state during local session
              restart";
          }
          enum REMOTE-HELPER {
            description
              "The local system is able to retain routes during restart
              but the remote system is only able to act as a helper";
          }
        }
```

```
        description
          "Ths leaf indicates the mode of operation of BGP graceful
          restart with the peer";
      }
    }

    grouping bgp-neighbor-afi-safi-graceful-restart_state {
      description
        "Operational state variables relating to the graceful-restart
        mechanism on a per-AFI-SAFI basis";

      leaf received {
        type boolean;
        description
          "This leaf indicates whether the neighbor advertised the
          ability to support graceful-restart for this AFI-SAFI";
      }

      leaf advertised {
        type boolean;
        description
          "This leaf indicates whether the ability to support
          graceful-restart has been advertised to the peer";
      }
    }


  }


  <CODE ENDS>
```

6. References

6.1. Normative references

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC1997] Chandrasekeran, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, August 1996.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, April 2006.

[RFC3065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 3065, February 2001.

[RFC2439] Villamizar, C., Chandra, R., and R. Govindan, "BGP Route Flap Damping", RFC 2439, November 1998.

[RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", RFC 4724, January 2007.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, January 2013.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

6.2. Informative references

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-00 (work in progress), March 2015.

[I-D.shaikh-rtgwg-policy-model]
Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-shaikh-rtgwg-policy-model-00 (work in progress), January 2015.

Appendix A. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Pavan Beeram, Chris Chase, Ed Crabbe, Luyuan Fang, Bill Fenner, Akshay Gattani, Josh George, Vijay Gill, Jeff Haas, Dhanendra Jain, Ina Minei, Carl Moberg, Ashok Narayanan, Einar Nilsen-Nygaard, Adam Simpson, Puneet Sood, Jason Sterne, Jim Uttaro, and Gunter Vandevelde.

Appendix B. Change summary

B.1. Changes between revisions -01 and -02

The -02 revision generally does not affect the structure of the model, but there are a number of changes that improve clarity, or simplify mappings to implementations.

- o Reorganized modules to move BGP-specific types, including policy-related types, to the bgp-types module.

- o Added ability to configure route selection options at the global level. Removed route selection options from per-neighbor AFI-SAFI since it is not widely implemented.

- o Removed a number of presence containers (e.g., in AFI-SAFI, graceful-restart, and multipath). Presence has been replaced with an explicit 'enabled' leaf to simplify semantics of inheritance and override in other parts of the BGP hieararchy.

- o Fixed inconsistencies with use-multiple-paths config. Added 'enabled' leaves with default false.

- o Updated top level list key to be a leafref in most lists for consistency with the operational state modeling approach outlined in [I-D.openconfig-netmod-opstate].

- o Some minor renaming / restructuring of routing policy defined-sets and apply-policy container.

- o Added ability to increment/decrement MED value in BGP policy statement actions.

- o Changed local-address leaf to be a union which can specify either an IP address, or be a reference to an interface.

- o Changed ebgp-multihop to add 'enabled' leaf and TTL behavior.

- o Changed BGP well-known communities from a enumeration to a identity. Also made consistent with IANA-defined standard well-known communities.

- o Remove enumeration values from BGP origin type given the disagreement in RFC 4271 and RFC 4273.

- o Added an additional match-options enumerated type for sets that only support ANY | INVERT, but not ALL behavior.

- o Modified prefix-sets to use inet:ip-prefix type and removed range checking on masklength-range (also added 'exact' specifier for masklength range).

- o Removed send-update-delay timer leaf. Desired behavior is reflected by minimum-advertisement-interval, i.e., MRAI.

- o Changed allow-own-as leaf to allow specification of a number of occurrences -- some implementations only support 0 or 1, i.e., equivalent of a boolean but several others support multiple occurrences.

B.2. Changes between revisions -00 and -01

The -01 revision reflects a number of changes, many based on feedback from implementors of the model on various routing platforms.

- o Refactored model to explicitly provide 'config' and 'state' containers at each leaf node to enable consistent and predictable access to operational state data corresponding to configuration data. This is based on the model design in [I-D.openconfig-netmod-opstate].

- o Refactored multiprotocol module with explicit set of supported AFI-SAFI combinations (using YANG identities) in a flattened list. Focus was on common config with more AFI-SAFI specific configuration forthcoming in future revisions.

- o Refactored BGP policy module to work with a new general routing policy model [I-D.shaikh-rtgwg-policy-model] by augmenting it with BGP-specific policy options (conditions, actions, and defined sets).

- o Added enclosing containers to lists (e.g., neighbors, peer-groups, and AFI-SAFI)

- o Removed neighbor configuration from the peer-group hierarchy. Neighbor configuration now has a peer-group leaf which references the peer group to which the neighbor belongs.

- o Several new configuration items added to base bgp module, including adding some configuration items to the global hierarchy level.

Authors' Addresses

Anees Shaikh (editor)
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: aashaikh@google.com

Rob Shakir (editor)
BT
pp. C3L, BT Centre
81, Newgate Street
London EC1A 7AJ
UK

Email: rob.shakir@bt.com
URI: http://www.bt.com/

Keyur Patel (editor)
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

Susan Hares (editor)
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
US

Email: kd6913@att.com

Deepak Bansal
Microsoft
205 108th Ave. NE, Suite 400
Bellevue, WA
US

Email: dbansal@microsoft.com

Alexander Clemm
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: alex@cisco.com

Aleksandr Zhdankin
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: azhdanki@cisco.com

Mahesh Jethanandani
Ciena
1741 Technology Drive
San Jose, CA 95110
USA

Email: mjethanandani@gmail.com

Xyfeng Liu
Ericsson
1595 Spring Hill Road, Suite 500
Vienna, VA 22182
USA

Email: xufeng.liu@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 5, 2016

A. Shaikh
Google
R. Shakir
BT
K. D'Souza
C. Chase
AT&T

July 4, 2015

Routing Policy Configuration Model for Service Provider Networks
draft-shaikh-rtgwg-policy-model-01

Abstract

This document defines a YANG data model for configuring and managing routing policies in a vendor-neutral way and based on actual operational practice. The model provides a generic policy framework which can be augmented with protocol-specific policy configuration.

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 http://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 January 5, 2016.

1. Introduction

This document describes a YANG [RFC6020] data model for routing policy configuration based on operational usage and best practices in a variety of service provider networks. The model is intended to be vendor-neutral, in order to allow operators to manage policy configuration in a consistent, intuitive way in heterogeneous environments with routers supplied by multiple vendors.

1.1. Goals and approach

This model does not aim to be feature complete -- it is a subset of the policy configuration parameters available in a variety of vendor implementations, but supports widely used constructs for managing how routes are imported, exported, and modified across different routing protocols. The model development approach has been to examine actual policy configurations in use across a number of operator networks. Hence the focus is on enabling policy configuration capabilities and structure that are in wide use.

Despite the differences in details of policy expressions and conventions in various vendor implementations, the model reflects the observation that a relatively simple condition- action approach can be readily mapped to several existing vendor implementations, and also gives operators an intuitive and straightforward way to express policy without sacrificing flexibility. A side affect of this design decision is that legacy methods for expressing policies are not considered. Such methods could be added as an augmentation to the model if needed.

Consistent with the goal to produce a data model that is vendor neutral, only policy expressions that are deemed to be widely available in existing major implementations are included in the model. Those configuration items that are only available from a single implementation are omitted from the model with the expectation they will be available in separate vendor-provided modules that augment the current model.

2. Model overview

The routing policy model is defined in two YANG modules, the main policy module, and an auxiliary module providing additional generic types. The model has three main parts:

- o A generic framework to express policies as sets of related conditions and actions. This includes match sets and actions that are useful across many routing protocols.

- o A structure that allows routing protocol models to add protocol-specific policy conditions and actions though YANG augmentations. There is a complete example of this for BGP [RFC4271] policies in the proposed vendor-neutral BGP data model [I-D.shaikh-idr-bgp-model].

- o A reusable grouping for attaching import and export rules in the context of routing configuration for different protocols, VRFs, etc. This also enables creation of policy chains and expressing default policy behavior.

These modules make use of the standard Internet types, such as IP addresses, autonomous system numbers, etc., defined in RFC 6991 [RFC6991].

3. Route policy expression

Policies are expressed as a sequence of top-level policy definitions each of which consists of a sequence of policy statements. Policy statements in turn consist of simple condition-action tuples. Conditions may include multiple match or comparison operations, and similarly, actions may effect multiple changes to route attributes, or indicate a final disposition of accepting or rejecting the route. This structure is shown below.

```
+--rw routing-policy
   +--rw policy-definitions
      +--rw policy-definition* [name]
         +--rw name          string
         +--rw statements
            +--rw statement* [name]
               +--rw name          string
               +--rw conditions
               |     ...
               +--rw actions
                     ...
```

3.1. Defined sets for policy matching

The models provides a set of generic sets that can be used for matching in policy conditions. These sets are applicable across multiple routing protocols, and may be further augmented by protocol-specific models which have their own defined sets. The supported defined sets include:

- o prefix sets - define a set of IP prefixes, each with an associated CIDR netmask range (or exact length)

- o neighbor sets - define a set of neighboring nodes by their IP addresses

- o tag set - define a set of generic tag values that can be used in matches for filtering routes

The model structure for defined sets is shown below.

```
  +--rw routing-policy
      +--rw defined-sets
         +--rw prefix-sets
         |  +--rw prefix-set* [prefix-set-name]
         |     +--rw prefix-set-name    string
         |     +--rw prefix* [ip-prefix masklength-range]
         |        +--rw ip-prefix           inet:ip-prefix
         |        +--rw masklength-range    string
         +--rw neighbor-sets
         |  +--rw neighbor-set* [neighbor-set-name]
         |     +--rw neighbor-set-name    string
         |     +--rw neighbor* [address]
         |        +--rw address    inet:ip-address
         +--rw tag-sets
            +--rw tag-set* [tag-set-name]
               +--rw tag-set-name    string
               +--rw tag* [value]
                  +--rw value    pt:tag-type
```

3.2. Policy conditions

Policy statements consist of a set of conditions and actions (either of which may be empty). Conditions are used to match route attributes against a defined set (e.g., a prefix set), or to compare attributes against a specific value.

Match conditions may be further modified using the match-set-options configuration which allows operators to change the behavior of a match. Three options are supported:

- o ALL - match is true only if the given value matches all members of the set.

- o ANY - match is true if the given value matches any member of the set.

o INVERT - match is true if the given value does not match any member of the given set.

Not all options are appropriate for matching against all defined sets (e.g., match ALL in a prefix set does not make sense). In the model, a restricted set of match options is used where applicable.

Comparison conditions may similarly use options to change how route attributes should be tested, e.g., for equality or inequality, against a given value.

While most policy conditions will be added by individual routing protocol models via augmentation, this routing policy model includes several generic match conditions and also the ability to test which protocol or mechanism installed a route (e.g., BGP, IGP, static, etc.). The conditions included in the model are shown below.

```
+--rw routing-policy
  +--rw policy-definitions
     +--rw policy-definition* [name]
        +--rw statements
           +--rw statement* [name]
              +--rw conditions
                 +--rw call-policy?
                 +--rw match-prefix-set!
                 |  +--rw prefix-set?
                 |  +--rw match-set-options?
                 +--rw match-neighbor-set!
                 |  +--rw neighbor-set?
                 |  +--rw match-set-options?
                 +--rw match-tag-set!
                 |  +--rw tag-set?
                 |  +--rw match-set-options?
                 +--rw install-protocol-eq?
                 +--rw igp-conditions
```

3.3. Policy actions

When policy conditions are satisfied, policy actions are used to set various attributes of the route being processed, or to indicate the final disposition of the route, i.e., accept or reject.

Similar to policy conditions, the routing policy model includes generic actions in addition to the basic route disposition actions. These are shown below.

```
+--rw routing-policy
  +--rw policy-definitions
     +--rw policy-definition* [name]
        +--rw statements
           +--rw statement* [name]
              +--rw actions
                 +--rw (route-disposition)?
                 |  +--:(accept-route)
                 |  |  +--rw accept-route?   empty
                 |  +--:(reject-route)
                 |     +--rw reject-route?   empty
                 +--rw igp-actions
                    +--rw set-tag?   pt:tag-type
```

3.4. Policy subroutines

Policy 'subroutines' (or nested policies) are supported by allowing policy statement conditions to reference other policy definitions using the call-policy configuration. Called policies apply their conditions and actions before returning to the calling policy statement and resuming evaluation. The outcome of the called policy affects the evaluation of the calling policy. If the called policy results in an accept-route (either explicit or by default), then the subroutine returns an effective boolean true value to the calling policy. For the calling policy, this is equivalent to a condition statement evaluating to a true value and evaluation of the policy continues (see Section 4). Note that the called policy may also modify attributes of the route in its action statements. Similarly, a reject-route action returns false and the calling policy evaluation will be affected accordingly.

Note that the called policy may itself call other policies (subject to implementation limitations). The model does not prescribe a nesting depth because this varies among implementations, with some major implementations only supporting a single subroutine, for example. As with any routing policy construction, care must be taken with nested policies to ensure that the effective return value results in the intended behavior. Nested policies are a convenience in many routing policy constructions but creating policies nested beyond a small number of levels (e.g., 2-3) should be discouraged.

4. Policy evaluation

Evaluation of each policy definition proceeds by evaluating its corresponding individual policy statements in order. When a condition statement in a policy statement is satisfied, the corresponding action statement is executed. If the action statement has either accept-route or reject-route actions, evaluation of the

current policy definition stops, and no further policy definitions in the chain are evaluated.

If the condition is not satisfied, then evaluation proceeds to the next policy statement. If none of the policy statement conditions are satisfied, then evaluation of the current policy definition stops, and the next policy definition in the chain is evaluated. When the end of the policy chain is reached, the default route disposition action is performed (i.e., reject-route unless an an alternate default action is specified for the chain).

5. Applying routing policy

Routing policy is applied by defining and attaching policy chains in various routing contexts. Policy chains are sequences of policy definitions (described in Section 3) that have an associated direction (import or export) with respect to the routing context in which they are defined. The routing policy model defines an apply-policy grouping that can be imported and used by other models. As shown below, it allows definition of import and export policy chains, as well as specifying the default route disposition to be used when no policy definition in the chain results in a final decision.

```
+--rw apply-policy
|  +--rw config
|  |  +--rw import-policy*
|  |  +--rw default-import-policy?   default-policy-type
|  |  +--rw export-policy*
|  |  +--rw default-export-policy?   default-policy-type
```

The default policy defined by the model is to reject the route for both import and export policies.

An example of using the apply-policy group in another routing model is shown below for BGP. Here, import and export policies are applied in the context of a particular BGP peer group. Note that the policy chains reference policy definitions by name that are defined in the routing policy model.

```
+--rw bgp!
    +--rw peer-groups
       +--rw peer-group* [peer-group-name]
          +--rw peer-group-name
          +--rw config
          |  +--rw peer-as?
          |  +--rw local-as?
          |  +--rw peer-type?
          |  +--rw auth-password?
          |  +--rw remove-private-as?
          |  +--rw route-flap-damping?
          |  +--rw send-community?
          |  +--rw description?
          |  +--rw peer-group-name?
          +--ro state
          |  +--ro peer-as?
          |  +--ro local-as?
          |  +--ro peer-type?
          |  +--ro auth-password?
          |  +--ro remove-private-as?
          |  +--ro route-flap-damping?
          |  +--ro send-community?
          |  +--ro description?
          |  +--ro peer-group-name?
          |  +--ro total-paths?
          |  +--ro total-prefixes?
          +--rw apply-policy
          |  +--rw config
          |  |  +--rw import-policy*
          |  |  +--rw default-import-policy?
          |  |  +--rw export-policy*
          |  |  +--rw default-export-policy?
          |  +--ro state
          |     +--ro import-policy*
          |     +--ro default-import-policy?
          |     +--ro export-policy*
          |     +--ro default-export-policy?
          ...
```

6. Routing protocol-specific policies

Routing models that require the ability to apply routing policy may augment the routing policy model with protocol or other specific policy configuration. The routing policy model assumes that additional defined sets, conditions, and actions may all be added by other models.

An example of this is shown below, in which the BGP configuration model in [I-D.shaikh-idr-bgp-model] adds new defined sets to match on community values or AS paths. The model similarly augments BGP-specific conditions and actions into the corresponding sections of the routing policy model.

```
+--rw routing-policy
   +--rw defined-sets
      +--rw prefix-sets
      |  +--rw prefix-set* [prefix-set-name]
      |     +--rw prefix-set-name
      |     +--rw prefix* [ip-prefix masklength-range]
      |        +--rw ip-prefix
      |        +--rw masklength-range
      +--rw neighbor-sets
      |  +--rw neighbor-set* [neighbor-set-name]
      |     +--rw neighbor-set-name
      |     +--rw neighbor* [address]
      |        +--rw address
      +--rw tag-sets
      |  +--rw tag-set* [tag-set-name]
      |     +--rw tag-set-name
      |     +--rw tag* [value]
      |        +--rw value
      +--rw bgp-pol:bgp-defined-sets
         +--rw bgp-pol:community-sets
         |  +--rw bgp-pol:community-set* [community-set-name]
         |     +--rw bgp-pol:community-set-name
         |     +--rw bgp-pol:community-member*
         +--rw bgp-pol:ext-community-sets
         |  +--rw bgp-pol:ext-community-set* [ext-community-set-name]
         |     +--rw bgp-pol:ext-community-set-name
         |     +--rw bgp-pol:ext-community-member*
         +--rw bgp-pol:as-path-sets
            +--rw bgp-pol:as-path-set* [as-path-set-name]
               +--rw bgp-pol:as-path-set-name
               +--rw bgp-pol:as-path-set-member*
```

7. Security Considerations

Routing policy configuration has a significant impact on network operations, and as such any related model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer configuration and operational data. Note that use of alternate transport or data encoding (e.g.,

JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

Most of the data elements in the policy model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

8. IANA Considerations

This YANG data model and the component modules currently use a temporary ad-hoc namespace. If and when it is placed on redirected for the standards track, an appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The routing policy YANG modules will be registered in the "YANG Module Names" registry [RFC6020].

9. YANG modules

The routing policy model is described by the YANG modules in the sections below.

9.1. Routing policy model

```
<CODE BEGINS> file routing-policy.yang
module routing-policy {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/routing-policy";

  prefix "rpol";

  // import some basic types
  import ietf-inet-types { prefix inet; }
  import policy-types {prefix pt; }


  // meta
  organization
    "OpenConfig working group";

  contact
    "OpenConfig working group
    netopenconfig@googlegroups.com";

  description
    "This module describes a YANG model for routing policy
```

configuration. It is a limited subset of all of the policy configuration parameters available in the variety of vendor implementations, but supports widely used constructs for managing how routes are imported, exported, and modified across different routing protocols. This module is intended to be used in conjunction with routing protocol configuration models (e.g., BGP) defined in other modules.

Route policy expression:

Policies are expressed as a set of top-level policy definitions, each of which consists of a sequence of policy statements. Policy statements consist of simple condition-action tuples. Conditions may include mutiple match or comparison operations, and similarly actions may be multitude of changes to route attributes or a final disposition of accepting or rejecting the route.

Route policy evaluation:

Policy definitions are referenced in routing protocol configurations using import and export configuration statements. The arguments are members of an ordered list of named policy definitions which comprise a policy chain, and optionally, an explicit default policy action (i.e., reject or accept).

Evaluation of each policy definition proceeds by evaluating its corresponding individual policy statements in order. When a condition statement in a policy statement is satisfied, the corresponding action statement is executed. If the action statement has either accept-route or reject-route actions, policy evaluation of the current policy definition stops, and no further policy definitions in the chain are evaluated.

If the condition is not satisfied, then evaluation proceeds to the next policy statement. If none of the policy statement conditions are satisfied, then evaluation of the current policy definition stops, and the next policy definition in the chain is evaluated. When the end of the policy chain is reached, the default route disposition action is performed (i.e., reject-route unless an an alternate default action is specified for the chain).

Policy 'subroutines' (or nested policies) are supported by allowing policy statement conditions to reference another policy definition which applies conditions and actions from the referenced policy before returning to the calling policy statement and resuming evaluation. If the called policy results in an accept-route (either explicit or by default), then

```
        the subroutine returns an effective true value to the calling
        policy.  Similarly, a reject-route action returns false.  If the
        subroutine returns true, the calling policy continues to evaluate
        the remaining conditions (using a modified route if the
        subroutine performed any changes to the route).";

      revision "2015-05-15" {
        description
          "Initial revision";
        reference "TBD";
      }


      // typedef statements

      typedef default-policy-type {
        type enumeration {
          enum ACCEPT-ROUTE {
            description "default policy to accept the route";
          }
          enum REJECT-ROUTE {
            description "default policy to reject the route";
          }
        }
        description "type used to specify default route disposition in
        a policy chain";
      }


      // grouping statements

      grouping generic-defined-sets {
        description
          "Data definitions for pre-defined sets of attributes used in
          policy match conditions.  These sets are generic and can
          be used in matching conditions in different routing
          protocols.";

        container prefix-sets {
          description
            "Enclosing container for defined prefix sets for matching";

          list prefix-set {
            key prefix-set-name;
            description
              "List of the defined prefix sets";

            leaf prefix-set-name {
```

```
            type string;
            description
              "name / label of the prefix set -- this is used to
              reference the set in match conditions";
          }

          list prefix {
            key "ip-prefix masklength-range";
            description
              "List of prefix expressions that are part of the set";

            leaf ip-prefix {
              type inet:ip-prefix;
              mandatory true;
              description
                "The prefix member in CIDR notation -- while the
                prefix may be either IPv4 or IPv6, most
                implementations require all members of the prefix set
                to be the same address family.  Mixing address types in
                the same prefix set is likely to cause an error.";
            }

            leaf masklength-range {
              type string {
                pattern '^([0-9]+\.\.[0-9]+)|exact$';
              }
              description
                "Defines a range for the masklength, or 'exact' if
                the prefix has an exact length.

                Example: 10.3.192.0/21 through 10.3.192.0/24 would be
                expressed as prefix: 10.3.192.0/21,
                masklength-range: 21..24.

                Example: 10.3.192.0/21 would be expressed as
                prefix: 10.3.192.0/21,
                masklength-range: exact";
            }
          }
        }
      }

      container neighbor-sets {
        description
          "Enclosing container for defined neighbor sets for matching";

        list neighbor-set {
          key neighbor-set-name;
```

```
          description
              "Definitions for neighbor sets";

          leaf neighbor-set-name {
            type string;
            description
                "name / label of the neighbor set -- this is used to
                reference the set in match conditions";
          }

          list neighbor {
            key "address";
            description
                "list of addresses that are part of the neighbor set";

            leaf address {
              type inet:ip-address;
              description
                  "IP address of the neighbor set member";
            }
          }
        }
      }

      container tag-sets {
        description
          "Enclosing container for defined tag sets for matching";

        list tag-set {
          key tag-set-name;
          description
            "Definitions for tag sets";

          leaf tag-set-name {
            type string;
            description
              "name / label of the tag set -- this is used to reference
              the set in match conditions";
          }

          list tag {
            key "value";
            description
              "list of tags that are part of the tag set";

            leaf value {
              type pt:tag-type;
              description
```

```
                 "Value of the tag set member";
            }
          }
        }
      }
    }

    grouping local-generic-conditions {
      description
          "Condition statement definitions for consideration of a local
          characteristic of a route";

      leaf install-protocol-eq {
        type identityref {
          base pt:install-protocol-type;
        }
        description
          "Condition to check the protocol / method used to install
          which installed the route into the local routing table";
      }
    }

    grouping match-set-options-group {
      description
        "Grouping containing options relating to how a particular set
        should be matched";

      leaf match-set-options {
        type pt:match-set-options-type;
        description
          "Optional parameter that governs the behaviour of the
          match operation";
      }
    }

    grouping match-set-options-restricted-group {
      description
        "Grouping for a restricted set of match operation modifiers";

      leaf match-set-options {
        type pt:match-set-options-restricted-type;
        description
          "Optional parameter that governs the behaviour of the
          match operation.  This leaf only supports matching on ANY
          member of the set or inverting the match.  Matching on ALL is
          not supported)";
      }
    }
```

```
     grouping generic-conditions {
       description "Condition statement definitions for checking
       membership in a generic defined set";

       container match-prefix-set {
         presence
           "The presence of this container indicates that the routes
           should match the prefix-set referenced.";

         description
           "Match a referenced prefix-set according to the logic
           defined in the match-set-options leaf";

         leaf prefix-set {
           type leafref {
             path "/routing-policy/defined-sets/prefix-sets/" +
             "prefix-set/prefix-set-name";
             //TODO: require-instance should be added when it's
             //supported in YANG 1.1
             //require-instance true;
           }
           description "References a defined prefix set";
         }
         uses match-set-options-restricted-group;
       }

       container match-neighbor-set {
         presence
           "The presence of this container indicates that the routes
           should match the neighbour set referenced";

         description
           "Match a referenced neighbor set according to the logic
           defined in the match-set-options-leaf";

         leaf neighbor-set {
           type leafref {
             path "/routing-policy/defined-sets/neighbor-sets/" +
             "neighbor-set/neighbor-set-name";
             //TODO: require-instance should be added when it's
             //supported in YANG 1.1
             //require-instance true;
           }
           description "References a defined neighbor set";
         }
         uses match-set-options-restricted-group;
       }
```

```
      container match-tag-set {
        presence
          "The presence of this container indicates that the routes
          should match the tag-set referenced";

        description
          "Match a referenced tag set according to the logic defined
          in the match-options-set leaf";

        leaf tag-set {
          type leafref {
            path "/routing-policy/defined-sets/tag-sets/tag-set" +
            "/tag-set-name";
            //TODO: require-instance should be added when it's
            //supported in YANG 1.1
            //require-instance true;
          }
          description "References a defined tag set";
        }
        uses match-set-options-restricted-group;
      }

      uses local-generic-conditions;
    }

    grouping igp-generic-conditions {
      description "grouping for IGP policy conditions";

    }


    grouping igp-conditions {
      description "grouping for IGP-specific policy conditions";

      container igp-conditions {
        description "Policy conditions for IGP attributes";

        uses igp-generic-conditions;

      }
    }

    grouping generic-actions {
      description
        "Definitions for common set of policy action statements that
        manage the disposition or control flow of the policy";

      choice route-disposition {
```

```
        description
          "Select the final disposition for the route, either
          accept or reject.";
        leaf accept-route {
          type empty;
          description "accepts the route into the routing table";
        }

        leaf reject-route {
          type empty;
          description "rejects the route";
        }
      }
    }


    grouping igp-actions {
      description "grouping for IGP-specific policy actions";

      container igp-actions {
        description "Actions to set IGP route attributes; these actions
        apply to multiple IGPs";

        leaf set-tag {
          type pt:tag-type;
          description
            "Set the tag value for OSPF or IS-IS routes.";
        }
      }
    }

    container routing-policy {
      description
        "top-level container for all routing policy configuration";

      container defined-sets {
        description
            "Predefined sets of attributes used in policy match
            statements";

        uses generic-defined-sets;
        // uses bgp-defined-sets;
        // don't see a need for IGP-specific defined sets at this point
        // e.g., for OSPF, IS-IS, etc.
      }

      container policy-definitions {
        description
```

```
          "Enclosing container for the list of top-level policy
          definitions";

        list policy-definition {

          key name;
          description
            "List of top-level policy definitions, keyed by unique
            name.  These policy definitions are expected to be
            referenced (by name) in policy chains specified in import/
            export configuration statements.";


          leaf name {
            type string;
            description
                "Name of the top-level policy definition -- this name
                  is used in references to the current policy";
          }

          container statements {
            description
              "Enclosing container for policy statements";

            list statement {
              key name;
              // TODO: names of policy statements within a policy defn
              // should be optional, however, YANG requires a unique id
              // for lists; not sure that a compound key works either;
              // need to investigate further.
              ordered-by user;
              description
                "Policy statements group conditions and actions within
                a policy definition.  They are evaluated in the order
                specified (see the description of policy evaluation
                at the top of this module.";

              leaf name {
                type string;
                description "name of the policy statement";
              }

              container conditions {

                description "Condition statements for this
                  policy statement";

                leaf call-policy {
```

```
                  type leafref {
                    path "/rpol:routing-policy/" +
                      "rpol:policy-definitions/" +
                      "rpol:policy-definition/rpol:name";
                    //TODO: require-instance should be added when it's
                    //supported in YANG 1.1
                    //require-instance true;
                  }
                  description
                    "Applies the statements from the specified policy
                    definition and then returns control the current
                    policy statement. Note that the called policy may
                    itself call other policies (subject to
                    implementation limitations). This is intended to
                    provide a policy 'subroutine' capability.  The
                    called policy should contain an explicit or a
                    default route disposition that returns an effective
                    true (accept-route) or false (reject-route),
                    otherwise the behavior may be ambiguous and
                    implementation dependent";
                }
                uses generic-conditions;
                uses igp-conditions;
              }

              container actions {

                description "Action statements for this policy
                      statement";

                uses generic-actions;
                uses igp-actions;
              }
            }
          }
        }
      }
    }

    grouping apply-policy-config {
      description
        "Configuration data for routing policies";

      leaf-list import-policy {
        type leafref {
          path "/rpol:routing-policy/rpol:policy-definitions/" +
            "rpol:policy-definition/rpol:name";
          //TODO: require-instance should be added when it's
```

```
          //supported in YANG 1.1
          //require-instance true;
        }
        ordered-by user;
        description
          "list of policy names in sequence to be applied on
          receiving a routing update in the current context, e.g.,
          for the current peer group, neighbor, address family,
          etc.";
      }

      leaf default-import-policy {
        type default-policy-type;
        default REJECT-ROUTE;
        description
          "explicitly set a default policy if no policy definition
          in the import policy chain is satisfied.";
      }

      leaf-list export-policy {
        type leafref {
          path "/rpol:routing-policy/rpol:policy-definitions/" +
            "rpol:policy-definition/rpol:name";
          //TODO: require-instance should be added when it's
          //supported in YANG 1.1
          //require-instance true;
        }
        ordered-by user;
        description
          "list of policy names in sequence to be applied on
          sending a routing update in the current context, e.g.,
          for the current peer group, neighbor, address family,
          etc.";
      }

      leaf default-export-policy {
        type default-policy-type;
        default REJECT-ROUTE;
        description
          "explicitly set a default policy if no policy definition
          in the export policy chain is satisfied.";
      }
    }

    grouping apply-policy-state {
      description
        "Operational state associated with routing policy";
```

```
      //TODO: identify additional state data beyond the intended
      //policy configuration.
    }

    grouping apply-policy-group {
      description
        "Top level container for routing policy applications. This
        grouping is intended to be used in routing models where
        needed.";

      container apply-policy {
        description
          "Anchor point for routing policies in the model.
          Import and export policies are with respect to the local
          routing table, i.e., export (send) and import (receive),
          depending on the context.";

        container config {
          description
            "Policy configuration data.";

          uses apply-policy-config;
        }

        container state {

          config false;
          description
            "Operational state for routing policy";

          uses apply-policy-config;
          uses apply-policy-state;
        }
      }
    }
  }
  <CODE ENDS>
```

9.2. Routing policy types

```
  <CODE BEGINS> file policy-types.yang
  module policy-types {

    yang-version "1";

    // namespace
    namespace "http://openconfig.net/yang/policy-types";
```

```
     prefix "ptypes";

     // import some basic types
     import ietf-yang-types { prefix yang; }


     // meta
     organization
       "OpenConfig working group";

     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "This module contains general data definitions for use in routing
       policy.  It can be imported by modules that contain protocol-
       specific policy conditions and actions.";

     revision "2015-05-15" {
       description
         "Initial revision";
       reference "TBD";
     }

     // identity statements

     identity attribute-comparison {
       description
         "base type for supported comparison operators on route
         attributes";
     }

     identity attribute-eq {
       base attribute-comparison;
       description "== comparison";
     }

     identity attribute-ge {
       base attribute-comparison;
       description ">= comparison";
     }

     identity attribute-le {
       base attribute-comparison;
       description "<= comparison";
     }
```

```
     typedef match-set-options-type {
       type enumeration {
         enum ANY {
           description "match is true if given value matches any member
           of the defined set";
         }
         enum ALL {
           description "match is true if given value matches all
           members of the defined set";
         }
         enum INVERT {
           description "match is true if given value does not match any
           member of the defined set";
         }
       }
       default ANY;
       description
         "Options that govern the behavior of a match statement.  The
         default behavior is ANY, i.e., the given value matches any
         of the members of the defined set";
     }

     typedef match-set-options-restricted-type {
       type enumeration {
         enum ANY {
           description "match is true if given value matches any member
           of the defined set";
         }
         enum INVERT {
           description "match is true if given value does not match any
           member of the defined set";
         }
       }
       default ANY;
       description
         "Options that govern the behavior of a match statement.  The
         default behavior is ANY, i.e., the given value matches any
         of the members of the defined set.  Note this type is a
         restricted version of the match-set-options-type.";
         //TODO: restriction on enumerated types is only allowed in
         //YANG 1.1.  Until then, we will require this additional type
     }

     grouping attribute-compare-operators {
       description "common definitions for comparison operations in
       condition statements";

       leaf operator {
```

```
          type identityref {
            base attribute-comparison;
          }
          description
            "type of comparison to be performed";
        }

      leaf value {
        type uint32;
        description
          "value to compare with the community count";
      }
    }

    typedef tag-type {
      type union {
        type uint32;
        type yang:hex-string;
      }
      description "type for expressing route tags on a local system,
      including IS-IS and OSPF; may be expressed as either decimal or
      hexidecimal integer";
      reference
        "RFC 2178 OSPF Version 2
        RFC 5130 A Policy Control Mechanism in IS-IS Using
        Administrative Tags";
    }

    identity install-protocol-type {
      description
        "Base type for protocols which can install prefixes into the
        RIB";
    }

    identity BGP {
      base install-protocol-type;
      description "BGP";
      reference "RFC 4271";
    }

    identity ISIS {
      base install-protocol-type;
      description "IS-IS";
      reference "ISO/IEC 10589";
    }

    identity OSPF {
      base install-protocol-type;
```

```
      description "OSPFv2";
      reference "RFC 2328";
    }

    identity OSPF3 {
      base install-protocol-type;
      description "OSPFv3";
      reference "RFC 5340";
    }

    identity STATIC {
      base install-protocol-type;
      description "Locally-installed static route";
    }

    identity DIRECTLY-CONNECTED {
      base install-protocol-type;
      description "A directly connected route";
    }

    identity LOCAL-AGGREGATE {
      base install-protocol-type;
      description "Locally defined aggregate route";
    }
  }
  <CODE ENDS>
```

10. Policy examples

Below we show an example of XML-encoded configuration data using the routing policy and BGP models to illustrate both how policies are defined, and also how they can be applied. Note that the XML has been simplified for readability.

```
    <routing-policy>

      <defined-sets>
        <prefix-sets>
          <prefix-set>
            <prefix-set-name>prefix-set-A</prefix-set-name>
            <prefix>
              <ip-prefix>192.0.2.0/24</ip-prefix>
              <masklength-range>24..32</masklength-range>
            </prefix>
            <prefix>
              <ip-prefix>10.0.0.0/16</ip-prefix>
```

```
              <masklength-range>16..32</masklength-range>
            </prefix>
            <prefix>
              <ip-prefix>192.168.0.0/19</ip-prefix>
              <masklength-range>19..24</masklength-range>
            </prefix>
          </prefix-set>
         </prefix-sets>
         <tag-sets>
          <tag-set>
           <tag-set-name>cust-tag1</tag-set-name>
           <tag>
            <value>10</value>
          </tag>
         </tag-set>
        </tag-sets>
      </defined-sets>

     <policy-definitions>
       <policy-definition>
         <name>export-tagged-BGP</name>
         <statements>
           <statement>
             <name>term-0</name>
             <conditions>
               <install-protocol-eq>OSPF3</install-protocol-eq>
               <match-tag-set>
                 <tag-set>cust-tag1</tag-set>
               </match-tag-set>
             </conditions>
             <actions>
               <accept-route />
             </actions>
           </statement>
         </statements>
       </policy-definition>
     </policy-definitions>

   </routing-policy>

   <bgp>
     <global>
       <config>
         <as>65517</as>
       </config>
     </global>
     <peer-groups>
       <peer-group>
```

```
        <peer-group-name>PG1</peer-group-name>
        <config>
          <peer-as>65518</peer-as>
          <peer-type>EXTERNAL</peer-type>
        </config>
        <apply-policy>
          <config>
            <export-policy>export-tagged-BGP</export-policy>
          </config>
        </apply-policy>
      </peer-group>
    </peer-groups>
  </bgp>
```

11. References

11.1. Normative references

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2014.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

[RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.

11.2. Informative references

[I-D.shaikh-idr-bgp-model] Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-shaikh-idr-bgp-model-02 (work in progress), June 2015.

[I-D.openconfig-netmod-opstate] Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-00 (work in progress), March 2015.

Appendix A. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Luyuan Fang, Josh George, Acee Lindem, Stephane Litkowski, Ina Minei, Carl Moberg, Eric Osborne, Steve Padgett, Juergen Schoenwaelder, Jim Uttaro, and Russ White.

Appendix B. Change summary

B.1. Changes between revisions -00 and -01

The -01 revision of the policy model reflects a number of changes to the data model based on additional operator, reviewer, and implementor feedback.

- o Modified the apply-policy container to use the pattern for modeling operational state described in [I-D.openconfig-netmod-opstate].

- o Updated prefix lists to use ip-prefix type and masklength range to better enable range-checking and validation. Added an 'exact' option to the masklength range.

- o Changed accept / reject route to be within a choice statement.

- o Added enclosing containers to lists.

- o Minor changes to leaf-lists in defined sets definitions; also to apply-policy container structure.

- o Added second type of match-options set to handle restricted case of only ANY | INVERT (i.e., without ALL). The restricted enumerated type is now associated with the appropriate types of sets.

- o Moved install-protocol-type identity to policy-types module.

- o Removed require-instance statements from leafrefs pending availability in YANG 1.1.

- o Fixed discrepancies in the example shown in the document, and simplified the example.

Authors' Addresses

Anees Shaikh
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: aashaikh@google.com

Rob Shakir
BT
pp. C3L, BT Centre
81, Newgate Street
London EC1A 7AJ
UK

Email: rob.shakir@bt.com
URI: http://www.bt.com/

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
US

Email: kd6913@att.com

Chris Chase
AT&T
9505 Arboretum Blvd
Austin, TX
US

Email: chase@labs.att.com

IDR and SIDR
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

K. Sriram
D. Montgomery
US NIST

March 9, 2015

Methods for Detection and Mitigation of BGP Route Leaks
draft-sriram-idr-route-leak-detection-mitigation-00

Abstract

In [I-D.ietf-grow-route-leak-problem-definition], the authors have provided a definition of the route leak problem, and also enumerated several types of route leaks. In this document, we first examine which of those route-leak types are detected and mitigated by the existing origin validation [RFC 6811] and BGPSEC path validation [I-D.ietf-sidr-bgpsec-protocol]. Where the current BGPSEC protocol doesn't offer a solution, this document suggests an enhancement that would extend the route-leak detection and mitigation capability of BGPSEC. The solution can be implemented in BGP without necessarily tying it to BGPSEC. Incorporating the solution in BGPSEC is one way of implementing it in a secure way. We do not claim to have provided a solution for all possible types of route leaks, but the solution covers several, especially considering some significant route-leak attacks or occurrences that have been observed in recent years. The document also includes a stopgap method for detection and mitigation of route leaks for the phase when BGPSEC (path validation) is not yet deployed but only origin validation is deployed.

Status of This Memo

Table of Contents

1. Introduction 2
2. Mechanisms for Detection and Mitigation of Route Leaks . . . 3
  2.1. Route Leak Protection (RLP) Field Encoding by Sending Router 5
  2.2. Recommended Actions at a Receiving Router 7
  2.3. Detection and Mitigation of Route Leaks of Type 5 and Type 6 8
3. Stopgap Solution when Only Origin Validation is Deployed . . 9
4. Design Rationale and Discussion 10
  4.1. Downside of 'Up (Towards Provider AS)' Indication in the RLP Field 10
  4.2. Possibility of Abuse of '01' (i.e. 'Do not Propagate Up') Indication in the RLP Field 10
5. Summary 11
6. Security Considerations 11
7. IANA Considerations 11
8. Acknowledgements 11
9. References 11
  9.1. Normative References 11
  9.2. Informative References 12
Authors' Addresses 14

1. Introduction

In [I-D.ietf-grow-route-leak-problem-definition], the authors have provided a definition of the route leak problem, and also enumerated several types of route leaks. In this document, we first examine which of those route-leak types are detected and mitigated by the existing Origin Validation (OV) [RFC6811] and BGPSEC path validation [I-D.ietf-sidr-bgpsec-protocol]. For the rest of this document, we use the term BGPSEC as synonymous with path validation. The BGPSEC

protocol provides cryptographic protection for some aspects of BGP update messages. OV and BGPSEC together offer mechanisms to protect against mis-originations and hijacks of IP prefixes as well as man-in-the-middle (MITM) AS path modifications. Route leaks (see [I-D.ietf-grow-route-leak-problem-definition] and references cited at the back) are another type of vulnerability in the global BGP routing system against which BGPSEC so far offers only partial protection.

For the types of route leaks enumerated in [I-D.ietf-grow-route-leak-problem-definition], where the current BGPSEC protocol doesn't offer a solution, this document suggests an enhancement that would extend the detection and mitigation capability of BGPSEC. The solution can be implemented in BGP without necessarily tying it to BGPSEC. Incorporating the solution in BGPSEC is one way of implementing it in a secure way. We do not claim to provide a solution for all possible types of route leaks, but the solution covers several relevant types, especially considering some significant route-leak occurrences that have been observed frequently in recent years. The document also includes (in Section 3) a stopgap method for detection and mitigation of route leaks for the phase when BGPSEC (path validation) is not yet deployed but only origin validation is deployed.

2. Mechanisms for Detection and Mitigation of Route Leaks

Referring to the enumeration of route leaks discussed in [I-D.ietf-grow-route-leak-problem-definition], Table 1 summarizes the route-leak detection capability offered by OV and BGPSEC for different types of route leaks. (Note: Route filtering is not considered here in this table. Please see Section 3.)

A detailed explanation of the contents of Table 1 is as follows. It is readily observed that route leaks of Types 1, 5, 6, and 7 are not detected by OV or even by BGPSEC. Type 2 route leak involves changing a prefix to a subprefix (i.e. more specific); such a modified update will fail BGPSEC checks. Clearly, Type 3 route leak involves hijacking and hence can be detected by OV. In the case of Type 3 route leak, there would be no existing ROAs to validate a re-originated prefix or subprefix, and hence the update will be considered Invalid by OV.

Type of Route Leak	Detection Coverage and Comments
Type 1: U-Turn with Full Prefix	Neither OV nor BGPSEC (in its current form) detects Type 1.
Type 2: U-Turn with More Specific Prefix	In OV, the ROA maxLength may offer detection of Type 2 in some cases; BGPSEC (in its current form) always detects Type 2.
Type 3: Prefix Hijack with Data Path to Legitimate Origin	OV by itself detects Type 3; BGPSEC does not detect Type 3.
Type 4: Leak of Internal Prefixes and Accidental Deaggregation	For internal prefixes never meant to be seen (i.e. routed) on the Internet, OV helps detect their leak; they might either have no covering ROA or have a ROA-AS0 to always filter them. In the case of accidental deaggregation, OV may offer some detection due to ROA maxLength. BGPSEC does not catch Type 4.
Type 5: Lateral ISP-ISP-ISP Leak	Neither OV nor BGPSEC (in its current form) detects Type 5.
Type 6: Leak of Provider Prefixes to Peer	Neither OV nor BGPSEC (in its current form) detects Type 6.
Type 7: Leak of Peer Prefixes to Provider	Neither OV nor BGPSEC (in its current form) detects Type 7.

Table 1: Examination of Route-Leak Detection Capability of Origin Validation and Current BGPSEC Path Validation

In the case of Type 4 leaks involving internal prefixes that are not meant to be routed in the Internet, they are likely to be detected by OV. That is because such prefixes might either have no covering ROA or have a ROA-AS0 to always filter them. In the case of Type 4 leaks that are due to accidental deaggregation, they may be detected due to violation of ROA maxLength. BGPSEC does not catch Type 4. However, route leaks of Type 4 are least problematic due to the following reasons. In the case of accidental deaggregation, the offending AS

is itself the legitimate destination of the leaked more-specific prefixes. Hence, in most cases of this type, the data traffic is neither misrouted not denied service. Also, leaked announcements of Type 4 are short-lived and typically withdrawn quickly following the announcements. Further, the MaxPrefix limit may kick in in some receiving routers and that helps limit the propagation of sometimes large number of leaked routes of Type 4.

From the above, it is evident that in our proposed solution method, we need to focus primarily on route leaks of Types 1, 5, 6, and 7. In Section 2.1 and Section 2.2, we describe a simple addition to BGPSEC that facilitates cryptographically-enabled detection of route leaks of Types 1 and 7. Then in Section 2.3, we will explain how the same method as described in Section 2.1 can be utilized between ISPs (or ASes) to detect and mitigate route leaks of Types 5 and 6.

2.1. Route Leak Protection (RLP) Field Encoding by Sending Router

The key principle is that, in the event of a route leak, a receiving router in a provider AS (e.g. referring to Figure 1, ISP2 (AS3) router) should be able to detect from the prefix-update that its customer AS (e.g. AS1 in Figure 1) SHOULD NOT have forwarded the update (towards the provider AS). This means that at least one of the ASes in the AS path of the update has indicated that it sent the update to its customer or peer AS, but forbade any subsequent 'Up' forwarding (i.e. from a customer AS to its provider AS). For this purpose, a Route Leak Protection (RLP) field to be set by a sending router is proposed to be used for each AS hop.

```
                                     /\              /\
                                      \ route-leak(P)/
                                       \ propagated /
                                        \          /
             +------------+    peer    +------------+
      ______| ISP1 (AS2) |----------->|  ISP2 (AS3)|---------->
     /       ------------+  prefix(P) +------------+ route-leak(P)
    | prefix |          \   update      /\        \  propagated
     \  (P)  /           \              /          \
      -------   prefix(P) \            /            \
                   update  \          /              \
                            \        /route-leak(P)  \/
                             \/      /
                         +---------------+
                         | customer(AS1) |
                         +---------------+
```

Figure 1: Illustration of the basic notion of a route leak.

For the purpose of route leak detection and mitigation proposed in this document, the RLP field value SHOULD be set to one of two values as follows:

- o 00: This is the default value (i.e. "nothing specified"),

- o 01: This is the 'Do not Propagate Up' indication; sender indicating that the prefix-update SHOULD NOT be forwarded 'Up' towards a provider AS.

There are two different scenarios when a sending AS SHOULD set the '01' indication in a prefix-update: (1) when sending the prefix-update to a customer AS, and (2) to let a peer AS know not to forward the prefix-update 'Up' towards a provide AS. In essence, in both scenarios, the intent of '01' indication is that any receiving AS along the subsequent AS path SHOULD NOT forward the prefix-update 'Up' towards its (receiving AS's) provider AS.

One may argue for an RLP field value (e.g. '10') to be used to specify 'Up' (i.e. towards provider AS) directionality. But in the interest of keeping the methodology simple, the choice of two RLP field values as defined above (00 - default, and 01 - 'Do not Propagate Up') is all that is needed. This two-state specification in the RLP field can be shown to work for detection and mitigation of route leaks of Types 1 and 7 readily (and also Types 5 and 6; see Section 2.3), which are the focus here. (Please see Section 4 for further discussion about the downside using 'Up' indication.)

In general, the proposed RLP encoding can be carried in BGP-4 [RFC4271] updates in any possible way, e.g., in a transitive community attribute. We consider BGPSEC as an example, where the RLP encoding can be accommodated in the existing Flags field and thereby secured using the existing BGPSEC path signatures. The Flags field is part of the Secure_Path Segment in BPGSEC updates [I-D.ietf-sidr-bgpsec-protocol]. It is one octet long, and one Flags field is available for each AS hop, and currently only the first bit is used in BGPSEC. So there are 7 bits that are currently unused in the Flags field. Two (or more if needed) of these bits can be designated for the RLP field. Since the BGPSEC protocol specification requires a sending AS to include the Flags field in the data that are signed over, the RLP field for each hop (assuming it would be part of the Flags field) will be protected under the sending AS's signature.

2.2. Recommended Actions at a Receiving Router

We provide here an example set of receiver actions that work to detect and mitigate route leaks of Types 1 and 7 (in particular). This example algorithm serves as a proof of concept. However, other receiver algorithms or procedures can be designed (based on the same sender specification as in Section 2.1) and may perform with greater efficacy, and are by no means excluded.

A recommended receiver algorithm for detecting a route leak is as follows:

A receiving BGPSEC router SHOULD mark an update as a Route-Leak if ALL of the following conditions hold true:

1. The update is received from a customer AS.

2. It is Valid in accordance with the BGPSEC protocol.

3. The update has the RLP field set to '01' (i.e. 'Do not Propagate Up') indication for one or more hops (excluding the most recent) in the AS path.

The reason for stating "excluding the most recent" in the above algorithm is as follows. The provider AS already knows that most recent hop in the update is from its customer AS to itself, and hence it does not need to rely on the RLP field value set by the customer for detection of route leaks. (See further discussion in Section 4.1.)

After applying the above detection algorithm, a receiving router may use any policy-based algorithm of its own choosing to mitigate any

detected route leaks. An example receiver algorithm for mitigating a route leak is as follows:

- o If an update from a customer AS is marked as a Route-Leak, then the receiving router SHOULD prefer a Valid signed update from a peer or an upstream provider over the customer's update.

The basic principle here is that the presence of '01' value in the RLP field corresponding to one or more AS hops in the AS path of an update coming from a customer AS informs a receiving router in a provider AS that a route leak is likely occurring. The provider AS then overrides the "prefer customer route" policy, and instead prefers a route learned from a peer or another upstream provider over the customer's route.

A receiving router expects the RLP field value for any hop in the AS path to be either 00 or 01. However, if a different value (say, 10 or 11) is found in the RLP field, then an error condition will get flagged, and any further action is TBD.

2.3. Detection and Mitigation of Route Leaks of Type 5 and Type 6

The sender and receiver actions described in Section 2.1 and Section 2.2 clearly help detect and mitigate route leaks of Types 1 and 7. With a slightly modified interpretation of the RLP encoding on the receiver side, they can be extended to detect lateral ISP-ISP-ISP route leaks (Type 5) as well as leaks of provider prefixes to peer (Type 6). A sending ISP router would set RLP field value to '01' indication towards a peer AS or a customer AS, following the same sender principles as described in Section 2.1.

A recommended receiver algorithm for an ISP to detect a route leak of either Type 5 or Type 6 is as follows:

A receiving BGPSEC router SHOULD mark an update as a Route-Leak if ALL of the following conditions hold true:

1. The update is received from a lateral ISP peer or a customer AS.

2. It is Valid in accordance with the BGPSEC protocol.

3. The update has the RLP field set to '01' indication for one or more hops (excluding the most recent) in the AS path.

In the above algorithm, the receiving AS interprets the '01' indication slightly strongly (i.e. stronger than in Section 2.2) to mean "the update SHOULD NOT have been propagated laterally to a peer ISP like me either". The rationale here is based on the fact that

settlement-free ISP peers accept only customer prefix-routes from each other. The receiving AS applies the logic that if a preceding AS (excluding the most recent) set '01' indication, it means that the update was sent to a peer or a customer by the (preceding) AS, and the update should not be traversing a lateral peer-to-peer link subsequently.

The receiver algorithm for mitigation is up to the discretion of the ISP. It may simply prefer another unmarked (i.e. not route-leak) update from a different peer or an upstream ISP over a marked update.

3. Stopgap Solution when Only Origin Validation is Deployed

During a phase when BGPSEC path validation has not yet been deployed but only origin validation has been deployed, it would be good have a stopgap solution for route leaks. The stopgap solution can be in the form of construction of a prefix filter list from ROAs. A suggested procedure for constructing such a list comprises of the following steps:

- ISP makes a list of all the ASes (Cust_AS_List) that are in its customer cone (ISP's own AS is also included in the list). (Some of the ASes in Cust_AS_List may be multi-homed to another ISP and that is OK.)

- ISP downloads from the RPKI repositories a complete list (Cust_ROA_List) of valid ROAs that contain any of the ASes in Cust_AS_List.

- ISP creates a list of all the prefixes (Cust_Prfx_List) that are contained in any of the ROAs in Cust_ROA_List.

- Cust_Prfx_List is the allowed list of prefixes that is permitted by the ISP's AS, and will be forwarded by the ISP to upstream ISPs, customers, and peers.

- Any prefix not in Cust_Prfx_List but announced by any of the ISP's customers is marked as a potential route leak. Then the ISP's router SHOULD prefer a Valid (i.e. valid according to origin validation) and 'not marked' update from a peer or an upstream provider over the customer's marked update for that prefix.

Special considerations with regard to the above procedure may be needed for DDoS mitigation service providers. They typically originate or announce a DDoS victim's prefix to their own ISP on a short notice during a DDoS emergency. Some provisions would need to be made for such cases, and they can be determined with the help of inputs from DDoS mitigation service providers.

4. Design Rationale and Discussion

In this section, we will try to provide design justifications for the methodology specified in Section 2, and also answer some anticipated questions.

4.1. Downside of 'Up (Towards Provider AS)' Indication in the RLP Field

As we have shown in Section 2, route leak detection and mitigation can be performed without the use of 'Up' (i.e. from customer AS to provider AS) indication in the RLP field. The detection and mitigation action should primarily occur at a provider AS's router just as soon as a leaked update is received from a customer AS. At that point, a provider AS can be fooled if it merely looks to see if an offending customer AS has set an 'Up' indication in the RLP field. This is so since a customer AS intent on leaking a route can deliberately set "Not Specified (00)" indication in order to misguide its provider AS. So it seems better that a provider AS figures out that the update is moving in the 'Up' direction based only on its own (configuration-based) knowledge that the update is coming from one of its customer ASes. An 'Up' indication (if it were allowed) can be also potentially misused. For example, an AS in the middle can determine that a '01' (i.e. 'Do not Propagate Up') value already exists on one of the preceding AS hops in a received update's AS path. Then, said AS in the middle can deliberately set its own RLP field to signal 'Up', in which case the update may be erroneously marked as a route leak by a subsequent AS if it concludes that there was a valley in the AS path of the update. So there appears to be some possibility of misuse of 'Up' indication, and hence we proposed not including it in the RLP specification in Section 2. However, other proposals, if any, that aim to beneficially use an 'Up' indication in the RLP field would be worth discussing.

4.2. Possibility of Abuse of '01' (i.e. 'Do not Propagate Up') Indication in the RLP Field

In reality, there appears to be no gain or incentive for an AS to falsely set its own RLP field to '01' (i.e. 'Do not Propagate Up') indication in an update that it originates or forwards. The purpose of a deliberate route leak by an AS is to attract traffic towards itself, but if the AS were to falsely set its own RLP field to '01' value, it would be effectively repelling some or all traffic away from itself for the prefix in question (see receiver algorithms in Section 2.2 and Section 2.3).

5. Summary

It should be emphasized once again that the proposed route-leak detection method using the RLP encoding is not intended to cover all forms of route leaks. However, we feel that the solution covers several important types of route leaks, especially considering some significant route-leak attacks or occurrences that have been frequently observed in recent years. The solution can be implemented in BGP without necessarily tying it to BGPSEC. Carrying the proposed RLP encoding in a transitive community attribute in BGP is another way, but in order to prevent abuse, the community attribute would require cryptographic protection. Incorporating the RLP encoding in the BGPSEC Flags field is one way of implementing it securely using an already existing protection mechanism provided in BGPSEC path signatures.

6. Security Considerations

The proposed Route Leak Protection (RLP) field requires cryptographic protection. Since it is proposed that the RLP field be included in the Flags field in the Secure_Path Segment in BPGSEC updates, the cryptographic security mechanisms in BGPSEC are expected to also apply to the RLP field. The reader is therefore directed to the security considerations provided in [I-D.ietf-sidr-bgpsec-protocol].

7. IANA Considerations

No updates to the registries are suggested by this document.

8. Acknowledgements

The authors wish to thank Danny McPherson and Eric Osterweil for discussions related to this work. Also, thanks are due to Jared Mauch, Jeff Haas, Warren Kumari, Brian Dickson, Amogh Dhamdhere, Jakob Heitz, Geoff Huston, Randy Bush, Ruediger Volk, Andrei Robachevsky, Chris Morrow, and Sandy Murphy for comments, suggestions, and critique. The authors are also thankful to Padma Krishnaswamy, Oliver Borchert, and Okhee Kim for their comments and review.

9. References

9.1. Normative References

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

9.2. Informative References

[Cowie2010]
Cowie, J., "China's 18 Minute Mystery", Dyn Research/ Renesys Blog, November 2010, <http://research.dyn.com/2010/11/ chinas-18-minute-mystery/>.

[Cowie2013]
Cowie, J., "The New Threat: Targeted Internet Traffic Misdirection", Dyn Research/Renesys Blog, November 2013, <http://research.dyn.com/2013/11/ mitm-internet-hijacking/>.

[Gao] Gao, L. and J. Rexford, "Stable Internet routing without global coordination", IEEE/ACM Transactions on Networking, December 2001, <http://www.cs.princeton.edu/~jrex/papers/ sigmetrics00.long.pdf>.

[Gill] Gill, P., Schapira, M., and S. Goldberg, "A Survey of Interdomain Routing Policies", ACM SIGCOMM Computer Communication Review, January 2014, <https://www.cs.bu.edu/~goldbe/papers/survey.pdf>.

[Giotsas] Giotsas, V. and S. Zhou, "Valley-free violation in Internet routing - Analysis based on BGP Community data", IEEE ICC 2012, June 2012, <http://www0.cs.ucl.ac.uk/staff/V.Giotsas/files/ giotsas.icc.2012.pdf>.

[Hiran] Hiran, R., Carlsson, N., and P. Gill, "Characterizing Large-scale Routing Anomalies: A Case Study of the China Telecom Incident", PAM 2013, March 2013, <http://www3.cs.stonybrook.edu/~phillipa/papers/ CTelecom.html>.

[Huston2012]
Huston, G., "Leaking Routes", March 2012, <http://labs.apnic.net/blabs/?p=139/>.

[Huston2014]
Huston, G., "What's so special about 512?", September 2014, <http://labs.apnic.net/blabs/?p=520/>.

[I-D.ietf-grow-route-leak-problem-definition]
Sriram, K., Montgomery, D., McPherson, D., and E. Osterweil, "Problem Definition and Classification of BGP Route Leaks", draft-ietf-grow-route-leak-problem-definition-00 (work in progress), February 2015.

[I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M., "BGPsec Protocol Specification", draft-ietf-sidr-bgpsec-protocol-11 (work in progress), January 2015.

[Kapela-Pilosov]
Pilosov, A. and T. Kapela, "Stealing the Internet: An Internet-Scale Man in the Middle Attack", DEFCON-16 Las Vegas, NV, USA, August 2008, <https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf/>.

[Khare] Khare, V., Ju, Q., and B. Zhang, "Concurrent Prefix Hijacks: Occurrence and Impacts", IMC 2012, Boston, MA, November 2012, <http://www.cs.arizona.edu/~bzhang/paper/12-imc-hijack.pdf/>.

[LRL] Khare, V., Ju, Q., and B. Zhang, "Large Route Leaks", Project web page, 2012, <http://nrl.cs.arizona.edu/projects/lsrl-events-from-2003-to-2009/>.

[Labovitz]
Labovitz, C., "Additional Discussion of the April China BGP Hijack Inciden", Arbor Networks IT Security Blog, November 2010, <http://www.arbornetworks.com/asert/2010/11/additional-discussion-of-the-april-china-bgp-hijack-incident/>.

[Luckie] Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., and kc. claffy, "AS Relationships, Customer Cones, and Validation", IMC 2013, October 2013, <http://www.caida.org/~amogh/papers/asrank-IMC13.pdf>.

[Madory] Madory, D., "Why Far-Flung Parts of the Internet Broke Today", Dyn Research/Renesys Blog, September 2014, <http://research.dyn.com/2014/09/why-the-internet-broke-today/>.

[Mauch] Mauch, J., "BGP Routing Leak Detection System", Project web page, 2014, <http://puck.nether.net/bgp/leakinfo.cgi/>.

[Mauch-nanog]
Mauch, J., "Detecting Routing Leaks by Counting", NANOG-41 Albuquerque, NM, USA, October 2007, <https://www.nanog.org/meetings/nanog41/presentations/mauch-lightning.pdf/>.

[Paseka] Paseka, T., "Why Google Went Offline Today and a Bit about How the Internet Works", CloudFare Blog, November 2012, <http://blog.cloudflare.com/why-google-went-offline-today-and-a-bit-about/>.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, January 2013.

[Toonk] Toonk, A., "What Caused Today's Internet Hiccup", August 2014, <http://www.bgpmon.net/what-caused-todays-internet-hiccup/>.

[Wijchers]
Wijchers, B. and B. Overeinder, "Quantitative Analysis of BGP Route Leaks", RIPE-69, November 2014, <https://ripe69.ripe.net/presentations/157-RIPE-69-Routing-WG.pdf>.

[Zmijewski]
Zmijewski, E., "Indonesia Hijacks the World", Dyn Research/Renesys Blog, April 2014, <http://research.dyn.com/2014/04/indonesia-hijacks-world/>.

Authors' Addresses

Kotikalapudi Sriram
US NIST

Email: ksriram@nist.gov

Doug Montgomery
US NIST

Email: dougm@nist.gov

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 30, 2015

A. Zhdankin
K. Patel
A. Clemm
Cisco
S. Hares
Huawei
M. Jethanandani
Ciena
X. Liu
Ericsson

January 26, 2015

Yang Data Model for BGP Protocol
draft-zhdankin-idr-bgp-cfg-00.txt

Abstract

This document defines a YANG data model that can be used to configure and manage BGP.

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 http://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 July 30, 2015.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Requirements Language 3
2. Definitions and Acronyms 4
3. The Design of the Core Routing Data Model 4
  3.1. Overview 4
  3.2. BGP Router Configuration 4
    3.2.1. AF Configuration 5
      3.2.1.1. AF Specific Protocol Configuration 7
      3.2.1.2. BGP Bestpath Configuration 8
      3.2.1.3. BGP Neighbor Configuration 8
      3.2.1.4. BGP Dampening 8
      3.2.1.5. BGP Route Aggregation 8
      3.2.1.6. BGP Redistribution 8
    3.2.2. BGP Neighbor Configuration 8
    3.2.3. BGP RPKI 11
  3.3. Prefix Lists 12
4. BGP Yang Module 12
5. IANA Considerations 42
6. Security Considerations 42
7. Acknowledgements 42
8. Contributors 42
9. References 42
  9.1. Normative References 42
  9.2. Informative References 43
Authors' Addresses 43

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage BGP. The data model is very comprehensive in scope, resulting in a very large module being defined. When contemplating whether it would be appropriate to introduce a data model of such a large scope, we decided that there would be value in particular because BGP defines such a rich set of features, which makes the problem arising from heterogeneity involved when managing these features quite pronounced. Also, there is very little information that is designated as "mandatory", leaving the decision which capabilities to actually support to product implementations.

There are several distinct parts of the data model. The first part, by far the largest, serves to configure and manage BGP itself. It defines a large set of control knobs for that purpose, as well as a few data nodes that can be used to monitor health and gather statistics. The second part, much smaller than the first, defines a data model for the configuration of AS-Path and prefix-based filter lists, in essence policies that define the exchange of BGP messages between BGP peers. Together they form a complete data model that serves as a framework for configuration and management of BGP protocol and its policies.

The YANG module defined in this document has all the common building blocks for BGP protocol namely: Neighbor List, Address Family specific Parameters, Protocol Bestpath specific Parameters, Prefix based Filter Lists, and AS-PATH based Filter Lists.

1.1. 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 RFC 2119 [RFC2119].

2. Definitions and Acronyms

AF: Address Family

AS: Autonomous System

BGP: Border Gateway Protocol

HTTP: Hyper-Text Transfer Protocol

JSON: JavaScript Object Notation

L2VPN: Layer 2 VPN

NETCONF: Network Configuration Protocol

NSAP: Network Service Access Point

ReST: Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

RPKI: Resource Public Key Infrastructure

RTFilter: Route Filter

VPN: Virtual Private Network

YANG: Data definition language for NETCONF

3. The Design of the Core Routing Data Model

3.1. Overview

The overall data model consists of two main components, each contained in its own separate container. Container "bgp-router" is used to configure and manage BGP itself. It is by far the largest part of the model. Container "prefix-lists" is used to configure BGP prefix lists. BGP prefix lists defines the rules and policies that helps BGP restrict information to share with which other nodes.

3.2. BGP Router Configuration

The overall structure of the "bgp-router" part of the model is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration data, "?" designates optional nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```
module: bgp
   +--rw bgp-router
   |  +--rw bgp-version?           string
   |  +--rw local-as-number?       uint32
   |  +--rw local-as-identifier?   inet:ip-address
   |  +--rw rpki-config
   |  |  .....
   |  +--rw router-id
   |  |  .....
   |  +--rw af-configuration
   |     .....
   +--rw bgp-neighbors
      .....
```

The key components of the "bgp-router" model concern the configuration of the BGP neighbors, of the Resource Public Key Infrastructure (RPKI), and of address families (AF). Each is defined in the following subsections.

3.2.1. AF Configuration

AF-configuration is used to configure and manage BGP configuration on an address family basis. BGP is designed to carry routing information for multiple different address families as specified in [RFC4760]. AF-Configuration is indexed by (router-AS, AFI, SAFI, VRFID) [RFC4760] and [RFC4364]. It contains any AF specific protocol configuration, BGP Bestpath configuration parameters,BGP neighbor configuration parameters, BGP dampening parameters, BGP route aggregation parameters, and any BGP policy configuration like redistribution.

The overall structure of the AF Configuration data model is depicted in the following diagram. As before, brackets enclose list keys, "rw" means configuration data, "?" designates optional nodes, parantheses indicate choices. The figure does not depict all definitions; it is intended to illustrate the overall model structure. Roughly speaking, address family configuration allows for separate configuration of IPv4, IPv6, L2VPN, NSAP, VPNv4 and VPNv6 address families, as well as route filters. Within each address family, you have additional substructure, for example, to distinguish between configuration of unicast and multicast.

```
module: bgp
   +--rw bgp-router
   |  .....
   |  +--rw af-configuration
   |     +--rw ipv4
   |     |  +--rw mdt
```

```
      |      |  | .....
      |      |  +--rw multicast
      |      |  |  +--rw bgp
      |      |  |  |  .....
      |      |  |  +--rw auto-summary?        boolean
      |      |  |  +--rw aggregate-address?   inet:ip-address
      |      |  |  +--rw distance?            uint8
      |      |  |  +--rw network?             inet:ip-address
      |      |  |  +--rw (protocol)?
      |      |  |  |  .....
      |      |  |  +--rw default-metric?      uint32
      |      |  +--rw unicast
      |      |  |  +--rw bgp
      |      |  |  |  .....
      |      |  |  +--rw auto-summary?          boolean
      |      |  |  +--rw aggregate-address?     inet:ip-address
      |      |  |  +--rw distance?              uint8
      |      |  |  +--rw network?               inet:ip-address
      |      |  |  +--rw (protocol)?
      |      |  |  |  .....
      |      |  |  +--rw number-of-path?        uint8
      |      |  |  +--rw ibgp-number-of-path?   uint8
      |      |  |  +--rw synchronization?       boolean
      |      |  +--rw mvpn
      |      |     +--rw bgp
      |      |     |  .....
      |      |     +--rw auto-summary?   boolean
      |      +--rw ipv6
      |      |  +--rw multicast
      |      |  |  +--rw bgp
      |      |  |  |  .....
      |      |  |  +--rw aggregate-address?   inet:ip-address
      |      |  |  +--rw distance?            uint8
      |      |  |  +--rw network?             inet:ip-address
      |      |  |  +--rw (protocol)?
      |      |  |     .....
      |      |  +--rw unicast
      |      |  |  +--rw bgp
      |      |  |  |  .....
      |      |  |  +--rw aggregate-address?     inet:ip-address
      |      |  |  +--rw distance?              uint8
      |      |  |  +--rw network?               inet:ip-address
      |      |  |  +--rw (protocol)?
      |      |  |  |  .....
      |      |  |  +--rw default-metric?        uint32
      |      |  |  +--rw number-of-path?        uint8
      |      |  |  +--rw ibgp-number-of-path?   uint8
      |      |  |  +--rw synchronization?       boolean
```

```
|      |  +--rw mvpn
|      |     .....
|      +--rw l2vpn
|      |  +--rw vpls
|      |     .....
|      +--rw nsap
|      |  +--rw unicast
|      |     +--rw bgp
|      |     |  .....
|      |     +--rw default-metric?        uint32
|      |     +--rw number-of-path?        uint8
|      |     +--rw ibgp-number-of-path?   uint8
|      |     +--rw network?               inet:ip-address
|      |     +--rw (protocol)?
|      |     |  .....
|      |     +--rw synchronization?       boolean
|      +--rw rtfilter
|      |  +--rw unicast
|      |     .....
|      +--rw vpnv4
|      |  +--rw unicast
|      |  |  +--rw bgp
|      |  |  |  .....
|      |  |  +--rw number-of-path?        uint8
|      |  |  +--rw ibgp-number-of-path?   uint8
|      |  +--rw multicast
|      |     +--rw bgp
|      |     |  .....
|      |     +--rw number-of-path?        uint8
|      |     +--rw ibgp-number-of-path?   uint8
|      +--rw vpnv6
|         +--rw unicast
|            +--rw bgp
|               .....
```

The key AF configuration components are described in the following subsections.

3.2.1.1. AF Specific Protocol Configuration

AF specific protocol configuration involves configuration of the parameters that are specific to a given AF. For instance, configuration parameters specific to the consistency checking between prefixes and labels are specific to address families that are enabled with Labels. Similarly redistribution of routes from other protocols is specific to Address Families that are supported in other protocols.

3.2.1.2. BGP Bestpath Configuration

BGP BestPath Configuration Parameters involves configuration of the parameters that influence the BGP Bestpath decision. For instance, the ignore-as-path command allows BGP process to ignore as-path length check. The ignore-routerid command allows BGP process to ignore routerid check. The ignore-igp-metric command allows BGP process to ignore igp metric check. The ignore-cost-community command allows BGP process to ignore cost communities. The MED related commands influence MED comparision in the BGP Bestpath decision.

3.2.1.3. BGP Neighbor Configuration

BGP Neighbor Configuration Parameters involves configuration of the parameters that are neighbor address family specific. These commands include neighbor capabilities, neighbor policies and any protocol related parameters that are specific to BGP neighbor.

3.2.1.4. BGP Dampening

BGP Dampening Parameters involves configuration of the parameters that influence BGP Route Dampening. These parameters allow enabling of Route Dampening on an address family level. The Dampening configuration also allows configuration of Dampening specific parameters like max suppress time, resuse threshold, half life, and the suppress threshold.

3.2.1.5. BGP Route Aggregation

BGP Route Aggregation Parameters involves configuration of the parameters that enables BGP Route Aggregation.

3.2.1.6. BGP Redistribution

BGP Route Redistribution Parameters involves configuration of the parameters that enables BGP Route Redistribution from and to the BGP protocol.

3.2.2. BGP Neighbor Configuration

Bgp-neighbor is used to configure and manage BGP neighbors. BGP neighbor configuration is indexed by af-configuration, neighbor address and neighbor-AS. It contains configuration for any policies that are configured for a neighbor on an inbound or an outbound, any transport related configuration parameters, any protocol related configuration parameters, and any protocol capabilities related configuration parameters.

BGP-neighbor-groups are used to configure and manage set of BGP neighbors with common configuration. BGP-neighbor-groups are index by af-configuration and group-name.

The following diagram depicts the overall structure of the BGP Neighbors subtree. Brackets enclose list keys, "rw" means configuration, "ro" operational state data, and "?" designates optional nodes. Parantheses enclose choice and case nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```
module: bgp
   + ....
   +--rw bgp-neighbors
   |  +--rw bgp-neighbor* [peer-address]
   |     +--rw peer-address                inet:ip-address
   |     +--rw remote-as                   uint32
   |     +--rw prefix-list?                prefix-list-ref
   |     +--rw default-action?             actions-enum
   |     +--rw af-specific-config
   |     |  +--rw ipv4
   |     |  |  +--rw mdt
   |     |  |  |  .....
   |     |  |  +--rw unicast
   |     |  |  |  .....
   |     |  |  +--rw multicast
   |     |  |  |  .....
   |     |  |  +--rw mvpn
   |     |  |     .....
   |     |  +--rw ipv6
   |     |  |  +--rw unicast
   |     |  |  |  .....
   |     |  |  +--rw multicast
   |     |  |  |  .....
   |     |  |  +--rw mvpn
   |     |  |     .....
   |     |  +--rw l2vpn
   |     |  |  +--rw evpn
   |     |  |  |  .....
   |     |  |  +--rw vpls
   |     |  |     .....
   |     |  +--rw nsap
   |     |  |  +--rw unicast
   |     |  |     .....
   |     |  +--rw rtfilter
   |     |  |  +--rw unicast
   |     |  |     .....
   |     |  +--rw vpnv4
```

```
      |      |  |  +--rw unicast
      |      |  |  |  .....
      |      |  |  +--rw multicast
      |      |  |     .....
      |      |  +--rw vpnv6
      |      |     +--rw unicast
      |      |     |  .....
      |      |     +--rw multicast
      |      |        .....
      |      +--rw session-open-mode?         enumeration
      |      +--rw send-buffer-size?          uint32
      |      +--rw receive-buffer-size?       uint32
      |      +--rw precedence?                enumeration
      |      +--rw tcp-mss?                   uint16
      |      +--rw ttl-security?              boolean
      |      +--rw hold-time?                 uint16
      |      +--rw keepalive-time?            uint16
      |      +--rw bgp-neighbor-state
      |      |  .....
      |      +--rw bgp-neighbor-statistics
      |         .....
      +--rw bgp-neighbor-groups
      |  +--rw bgp-neighbor-group* [nbr-grp-name]
      |     +--rw nbr-grp-name          string
      |     +--rw remote-as             uint32
      |     +--rw prefix-list?          prefix-list-ref
      |     +--rw default-action?       actions-enum
      |     +--rw af-specific-config
      |     |  +--rw ipv4
      |     |  |  +--rw mdt
      |     |  |  |  .....
      |     |  |  +--rw unicast
      |     |  |  |  .....
      |     |  |  +--rw multicast
      |     |  |  |  .....
      |     |  |  +--rw mvpn
      |     |  |     .....
      |     |  +--rw ipv6
      |     |  |  +--rw unicast
      |     |  |  |  .....
      |     |  |  +--rw multicast
      |     |  |  |  .....
      |     |  |  +--rw mvpn
      |     |  |     .....
      |     |  +--rw l2vpn
      |     |  |  +--rw evpn
      |     |  |  |  .....
      |     |  |  +--rw vpls
```

```
      |     |  |     .....
      |     |  +--rw nsap
      |     |  |  +--rw unicast
      |     |  |     .....
      |     |  +--rw rtfilter
      |     |  |  +--rw unicast
      |     |  |     .....
      |     |  +--rw vpnv4
      |     |  |  +--rw unicast
      |     |  |  |  .....
      |     |  |  +--rw multicast
      |     |  |     .....
      |     |  +--rw vpnv6
      |     |     +--rw unicast
      |     |     |  .....
      |     |     +--rw multicast
      |     |        .....
      |     +--rw hold-time?            uint16
      |     +--rw keepalive-time?       uint16
```

3.2.3. BGP RPKI

rpki-config is used to configure and manage BGP Origin Validation. This feature is specific to IPv4 and IPv6 Address Families. It is indexed by af-configuration. It contains the configuration commands for the BGP RPKI Server, RPKI RTR Protocol and the BGP protocol. This includes configuration for the Server address, Server preference, RPKI RTR protocol specific parameters, choice of a transport for RPKI RTR Protocol, and BGP specific parameters including enabling and disabling of this feature for IBGP and EBGP routes.

The structure of the RPKI configuration data model is depicted below, per the same conventions used in the earlier diagrams.

```
module: bgp
   +--rw bgp-router
   |  .....
   |  +--rw rpki-config
   |  |  +--rw cache-server-config
   |  |  |  .....
   |  |  +--rw validation-config
   |  |  |  .....
   |  |  +--rw bestpath-computation
   |  |     .....
```

3.3. Prefix Lists

BGP Prefix Lists are used to manipulate Prefix information carried within a BGP. The prefix information carried within BGP is filtered or allowed using BGP Prefix Lists. BGP Prefix Lists consists of an ordered set of one or more rules that describe IPv4 or IPv6 prefixes range and an associated action rule that describes whether the matching prefixes should be dropped or permitted. The Prefix Lists are usually applied to a BGP neighbor as part of an inbound policy (applied to prefixes received by a neighbor) or an outbound policy (applied to prefixes sent by a neighbor).

The structure of the prefix list configuration data model is depicted below, per the same conventions used in the earlier diagrams.

```
module: bgp
   .....
   +--rw prefix-lists
      +--rw prefix-list [prefix-list-name]
         +--rw prefix-list-name    string
         +--rw prefixes
            +--rw prefix [seq-nr]
               +--rw seq-nr           uint16
               +--rw prefix-filter
                  +--rw (ip-address-group)?
                  |  .....
                  +--rw action             actions-enum
                  +--rw statistics
                     .....
```

Prefix lists are defined in a list in a designated container. Each prefix list in turn contains a list of prefixes, indexed by a sequency number. Each prefix is comprised of a prefix filter, used to match BGP packets, an action that is applied when a filter matches, and a set of statistics that indicate how often individual prefixes are applied.

4. BGP Yang Module

<CODE BEGINS> file "bgp@2013-07-15.yang"

```
module bgp {
  namespace "urn:cisco:params:xml:ns:yang:bgp";
  // replace with IANA namespace when assigned
  prefix bgp;

  import ietf-inet-types {
```

```
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-routing {
    prefix routing;
    revision-date 2014-11-10;
  }

  organization
    "Cisco Systems
     170 West Tasman Drive
     San Jose, CA 95134-1706
     USA";
  contact
    "Alexander Clemm alex@cisco.com
     Keyur Patel keyupate@cisco.com
     Aleksandr Zhdankin azhdanki@cisco.com";
  description
    "This YANG module defines the generic configuration
     data for BGP, which is common across all of the vendor
     implementations of the protocol. It is intended that the module
     will be extended by vendors to define vendor-specific
     BGP configuration parameters and policies,
     for example route maps or route policies.


     Terms and Acronyms

     BGP (bgp): Border Gateway Protocol

     IP (ip): Internet Protocol

     IPv4 (ipv4):Internet Protocol Version 4

     IPv6 (ipv6): Internet Protocol Version 6

     MED(med): Multi Exit Discriminator

     IGP (igp): Interior Gateway Protocol

     MTU (mtu) Maximum Transmission Unit
    ";

  revision 2015-01-14 {
    description
      "Initial revision.";
```

```
  }

  identity bgp-routing-protocol {
    base routing:routing-protocol;
    description
      "This identity represents BGP routing protocol.";
  }

  typedef prefix-list-ref {
    description
      "A reference to the prefix list which a bgp-neighbor can use.";
    type leafref {
      path "/routing:routing/routing:routing-instance/routing:routing-protocols/
routing:routing-protocol/bgp:bgp-routing/bgp:prefix-lists/bgp:prefix-list/bgp:pr
efix-list-name";
    }
  }

  typedef bgp-peer-admin-status {
    description
      "Administartive status of a BGP peer.";
    type enumeration {
      enum "unknown";
      enum "up";
      enum "down";
    }
  }

  typedef actions-enum {
    description
      "Permit/deny action.";
    type enumeration {
      enum "permit";
      enum "deny";
    }
  }

  grouping ACTIONS {
    description
      "Permit/deny action.";
    leaf action {
      type actions-enum;
      mandatory true;
    }
  }

  grouping slow-peer-config {
    description
      "Configure a slow-peer.";
    container detection {
```

```
      leaf enable {
        type boolean;
        default "true";
      }
      leaf threshold {
        type uint16 {
          range "120..3600";
        }
      }
    }
    leaf split-update-group {
      type enumeration {
        enum "dynamic";
        enum "static";
      }
    }
  }

  grouping update-group-management {
    description
      "Manage peers in BGP update group.";
    leaf split-as-override {
      description
        "Keeps peers with as-override in different update groups.";
      type boolean;
    }
  }

  grouping neighbour-base-af-config {
    description
      "A set of configuration parameters that is applicable to all neighbour add
ress families.";
    leaf active {
      description
        "Enable the address family for this neighbor.";
      type boolean;
      default "false";
    }
    leaf advertisement-interval {
      description
        "Minimum interval between sending BGP routing updates.";
      type uint32;
    }
    leaf allowas-in {
      description
        "Accept as-path with my AS present in it.";
      type boolean;
      default "false";
    }
```

```
    leaf maximum-prefix {
      description
        "Maximum number of prefixes accepted from this peer.";
      type uint32;
    }
    leaf next-hop-self {
      description
        "Enable the next hop calculation for this neighbor.";
      type boolean;
      default "true";
    }
    leaf next-hop-unchanged {
      description
        "Propagate next hop unchanged for iBGP paths to this neighbour.";
      type boolean;
      default "true";
    }

    container remove-private-as {
      leaf remove-private-as-number {
        description
          "Remove private AS number from outbound updates.";
        type boolean;
      }
      leaf replace-with-local-as {
        description
          "Replace private AS number with local AS.";
        type boolean;
      }
    }
    leaf route-reflector-client {
      description
        "Configure a neighbor as Route Reflector client.";
      type boolean;
      default "false";
    }
    leaf send-community {
      description
        "Send Community attribute to this neighbor.";
      type enumeration {
        enum "both";
        enum "extended";
        enum "standard";
      }
      default "standard";
    }
    uses slow-peer-config;
    leaf soo {
```

```
      description
        "Site-of-Origin extended community. Format is ASN:nn or IP-address:nn";
      type string;
    }
    leaf weight {
      description
        "Set default weight for routes from this neighbor.";
      type uint16;
    }
  }

  grouping neighbour-common-af-config {
    description
      "A set of configuration parameters that is applicable to all neighbour add
ress families,
       except of nsap and rtfilter.";
    uses neighbour-base-af-config;
    leaf prefix-list {
      description
        "Reference to the prefix list of this neighbour.";
      type prefix-list-ref;
    }
    leaf soft-reconfiguration {
      description
        "Allow inbound soft reconfiguration.";
      type boolean;
    }
  }

  grouping neighbour-cast-af-config {
    description
      "A set of configuration parameters that is applicable to both unicast and
multicast sub-address families.";
    uses neighbour-common-af-config;
    leaf propagate-dmzlink-bw {
      description
        "Propagate the DMZ link bandwidth.";
      type boolean;
    }
    container default-originate {
      description
        "Originate default route to this neighbor.";
      leaf enable {
        type boolean;
        default "false";
      }
    }
  }

  grouping neighbour-ip-multicast-af-config {
```

```
    description
      "A set of configuration parameters that is applicable to ip multicast.";
    uses neighbour-cast-af-config;
    leaf route-server-client-context {
      description
        "Specifies Route Server client context name.";
      type string;
    }
  }

  grouping neighbour-ip-unicast-af-config {
    description
      "A set of configuration parameters that is applicable to ip unicast.
       This grouping is intended to be extended by vendors as necessary to descr
ibe the vendor-specific configuration parameters.";
    uses neighbour-ip-multicast-af-config;
  }

  grouping bgp-af-config {
    description
      "A set of configuration parameters that is applicable to all address famil
ies of the BFP router.";
    leaf additional-paths {
      description
        "Additional paths in the BGP table.";
      type enumeration {
        enum "all";
        enum "best-n";
        enum "group-best";
      }
    }
    leaf advertise-best-external {
      description
        "Advertise best external path to internal peers.";
      type boolean;
    }
    container aggregate-timer {
      description
        "Configure aggregation timer.";
      leaf enable {
        type boolean;
        default "true";
      }
      leaf threshold {
        type uint16 {
          range "6..60";
        }
      }
    }
    container bestpath {
```

```
      description
        "Change the default bestpath selection.";
      choice bestpath-selection {
        case as-path {
          description
            "Configures a BGP routrer to not consider the autonomous system (AS)
 path during best path route selection.";
          leaf ignore-as-path {
            type boolean;
            default "false";
          }
        }
        case compare-routerid {
          description
            "Configures a BGP routrer to compare identical routes received from
different external peers
             during the best path selection process and to select the route with
 the lowest router ID as the best path.";
          leaf ignore-routerid {
            type boolean;
            default "false";
          }
        }
        case cost-community {
          description
            "Configures a BGP router to not evaluate the cost community attribut
e
             during the best path selection process.";
          leaf ignore-cost-community {
            type boolean;
            default "false";
          }
        }
        case igp-metric {
          description
            "Configures the system to ignore the IGP metric during BGP best path
 selection.";
          leaf ignore-igp-metric {
            type boolean;
            default "false";
          }
        }
        case mad-confed {
          description
            "Configure a BGP routing process to compare the Multi Exit Discrimin
ator (MED)
             between paths learned from confederation peers.";
          leaf enable {
            type boolean;
            default "false";
          }
          leaf missing-as-worst {
            description
              "Assigns a value of infinity to routes that are missing
```

```
               the Multi Exit Discriminator (MED) attribute,
               making the path without a MED value the least desirable path";
            type boolean;
            default "false";
          }
        }
      }
    }
    uses bgp-dampening;
    leaf propagate-dmzlink-bw {
      description
        "Use DMZ Link Bandwidth as weight for BGP multipaths.";
      type boolean;
    }
    leaf redistribute-internal {
      description
        "Allow redistribution of iBGP into IGPs (dangerous)";
      type boolean;
    }
    leaf scan-time {
      description
        "Configure background scanner interval in seconds.";
      type uint8 {
        range "5..60";
      }
    }
    uses slow-peer-config;
    leaf soft-reconfig-backup {
      description
        "Use soft-reconfiguration inbound only when route-refresh is not negotia
ted.";
      type boolean;
    }
  }

  grouping bgp-af-vpn-config {
    description
      "A set of configuration parameters that is applicable to vpn sub-address f
amily on the BGP router.";
    uses bgp-af-config;
    uses update-group-management;
  }

  grouping bgp-af-mvpn-config {
    description
      "A set of configuration parameters that is applicable to mvpn sub-address
family on the BGP router.";
    leaf scan-time {
      description
        "Configure background scanner interval in seconds.";
      type uint8 {
```

```
        range "5..60";
      }
    }
    uses slow-peer-config;
    leaf soft-reconfig-backup {
      description
        "Use soft-reconfiguration inbound only when route-refresh is not negotia
ted.";
      type boolean;
    }
    leaf propagate-dmzlink-bw {
      description
        "Use DMZ Link Bandwidth as weight for BGP multipaths.";
      type boolean;
    }
    leaf rr-group {
      description
        "Extended community list name.";
      type string;
    }
    uses update-group-management;
  }

  grouping redistribute {
    description
      "Redistribute information from another routing protocol.
       This grouping is intended to be augmented by vendors to implement vendor-
specific protocol redistribution configuration options.";
    choice protocol {
      case bgp {
        leaf enable-bgp {
          type boolean;
        }
      }
      case ospf {
        leaf enable-ospf {
          type boolean;
        }
      }
      case isis {
        leaf enable-isis {
          type boolean;
        }
      }
      case connected {
        leaf enable-connected {
          type boolean;
        }
      }
      case eigrp {
```

```
        leaf enable-eigrp {
          type boolean;
        }
      }
      case mobile {
        leaf enable-mobile {
          type boolean;
        }
      }
      case static {
        leaf enable-static {
          type boolean;
        }
      }
      case rip {
        leaf enable-rip {
          type boolean;
        }
      }
    }
  }

  grouping router-af-config {
    description
      "A set of configuration parameters that is applicable to all address famil
ies on the BGP router.";
    leaf aggregate-address {
      description
        "Configure BGP aggregate address.";
      type inet:ip-address;
    }
    leaf distance {
      description
        "Define an administrative distance.";
      type uint8 {
        range "1..255";
      }
    }
    leaf network {
      description
        "Specify a network to announce via BGP.";
      type inet:ip-address;
    }
    uses redistribute;
  }

  grouping maximum-paths {
    description
      "Configures packet forwarding over multiple paths.";
```

```
    leaf number-of-path {
      type uint8 {
        range "1..32";
      }
    }
    leaf ibgp-number-of-path {
      type uint8 {
        range "1..32";
      }
    }
  }
  grouping bgp-neighbor-config {
    leaf remote-as {
      type uint32;
        mandatory true;
    }
    leaf prefix-list {
      type prefix-list-ref;
    }
    leaf default-action {
      type actions-enum;
    }

    leaf neighbor-group-name {
      description
        "Neighbor group name.";
      type string;
    }

    container af-specific-config {
      description
      "Address family specific configuration parameters for the neighbours.";
      container ipv4 {
        container mdt {
          uses neighbour-common-af-config;
        }
        container unicast {
          uses neighbour-ip-unicast-af-config;
        }
        container multicast {
          uses neighbour-ip-multicast-af-config;
        }
        container mvpn {
          uses neighbour-cast-af-config;
        }
      }
      container ipv6 {
        container unicast {
```

```
          uses neighbour-ip-unicast-af-config;
        }
        container multicast {
          uses neighbour-ip-multicast-af-config;
        }
        container mvpn {
          uses neighbour-common-af-config;
        }
      }
      container l2vpn {
        container evpn {
          uses neighbour-common-af-config;
        }
        container vpls {
          uses neighbour-common-af-config;
        }
      }
      container nsap {
        container unicast {
          uses neighbour-base-af-config;
          leaf prefix-list {
            type prefix-list-ref;
          }
        }
      }
      container rtfilter {
        container unicast {
          uses neighbour-base-af-config;
          leaf soft-reconfiguration {
            description
            "Allow inbound soft reconfiguration.";
            type boolean;
          }
        }
      }
      container vpnv4 {
        container unicast {
           uses neighbour-cast-af-config;
        }
        container multicast {
           uses neighbour-cast-af-config;
        }
      }
      container vpnv6 {
        container unicast {
          uses neighbour-cast-af-config;
        }
        container multicast {
```

```
          uses neighbour-cast-af-config;
        }
      }
    }
  }

  grouping bgp-neighbor-transport-config {
    leaf session-open-mode {
      description
      "Establish neighbor session using TCP Open mode.";
      type enumeration {
        enum "active";
        enum "passive";
      }
    }

    leaf send-buffer-size {
      description
      "Set socket BGP send buffer size.";
        type uint32;
    }

    leaf receive-buffer-size {
      description
      "Receive socket BGP send buffer size.";
        type uint32;
    }

    leaf precedence {
      description
      "Set Precedence.";
        type enumeration {
          enum "routine";
          enum "immediate";
          enum "flash";
          enum "flash-override";
          enum "critical";
          enum "internet";
          enum "network";
        }
    }

    leaf tcp-mss {
      description
      "TCP MSS.";
        type uint16;
    }
```

```
    leaf ttl-security {
      description
      "TTL Security.";
        type boolean;
    }
  }

  grouping bgp-neighbor-timers {
    leaf hold-time {
      description
      "BGP Hold Time interval.";
      default 180;
      type uint16;
    }

    leaf keepalive-time {
      description
      "BGP Keepalive Time interval.";
      default 60;
      type uint16;
    }
  }

  grouping bgp-dampening {
    container bgp-dampening-params {
      description
      "BGP Route Flap Dampening.";

       leaf half-time {
         description
         "Half Time for the penalty.";
         type uint8 {
           range "1..45";
         }
      }

       leaf Reuse-time {
         description
         "Reuse Time.";
         type uint16 {
           range "1..20000";
         }
      }

       leaf supresss-time {
         description
         "Supress Time.";
         type uint16 {
```

```
          range "1..20000";
        }
      }

       leaf max-supress-time {
         description
         "Max Supress Time";
         type uint8 {
           range "1..255";
         }
      }
    }
  }
  augment "/routing:routing/routing:routing-instance/routing:routing-protocols/r
outing:routing-protocol" {
    container bgp-routing {
      description
        "BGP routing configuration";
      must "/routing:routing/routing:routing-instance/routing:routing-protocols/
routing:routing-protocol/routing:type = bgp-routing-protocol";
      container bgp-router {
        description
          "This is a top-level container for the BGP router.";
        leaf bgp-version {
          type string;
        }
        leaf local-as-number {
          type uint32;
        }
        leaf local-as-identifier {
          type inet:ip-address;
        }
        container router-id {
          description
          "Configures a fixed router ID for the local BGP routing process.";
          leaf enable {
            type boolean;
          }
          choice config-type {
            case static {
              leaf ip-address {
                type boolean;
              }
            }
            case auto-config {
              leaf enable-auto-config {
                type boolean;
              }
            }
          }
```

```
        }

        container rpki-config {
          description
            "RPKI configuration parameters.";
          container cache-server-config {
            description
              "Configure the RPKI cache-server parameters in rpki-server configu
ration mode.";
            choice server {
              case ip-address {
                leaf ip-address {
                  type inet:ip-address;
                  mandatory true;
                }
              }
              case host-name {
                leaf ip-host-address {
                  type inet:host;
                  mandatory true;
                }
              }
            }
            choice transport {
              description
                "Specifies a transport method for the RPKI cache.";
              case tcp {
                leaf tcp-port {
                  type uint32;
                }
              }
              case ssh {
                leaf ssh-port {
                  type uint32;
                }
              }
            }
            leaf user-name {
              type string;
            }
            leaf password {
              type string;
            }
            leaf preference-value {
              description
                "Specifies a preference value for the RPKI cache.
                 Setting a lower preference value is better.";
              type uint8 {
                range "1..10";
```

```
              }
            }
            leaf purge-time {
              description
                "Configures the time BGP waits to keep routes from a cache after
 the cache session drops. Set purge time in seconds.";
              type uint16 {
                range "30..360";
              }
            }
            choice refresh-time {
              description
                "Configures the time BGP waits in between sending periodic seria
l queries to the cache. Set refresh-time in seconds.";
              case disable {
                leaf refresh-time-disable {
                  type boolean;
                }
              }
              case set-time {
                leaf refresh-interval {
                  type uint16 {
                    range "15..3600";
                  }
                }
              }
            }
            choice response-time {
              description
                "Configures the time BGP waits for a response after sending a se
rial or reset query. Set response-time in seconds.";
              case disable {
                leaf response-time-disable {
                  type boolean;
                }
              }
              case set-time {
                leaf response-interval {
                  type uint16 {
                    range "15..3600";
                  }
                }
              }
            }
          }
          container validation-config {
            description
              "Controls the behavior of RPKI prefix validation processing.";
            leaf enable {
              description
                "Enables RPKI origin-AS validation.";
```

```
              type boolean;
              default "true";
            }
            leaf enable-ibgp {
              description
                "Enables the iBGP signaling of validity state through an extende
d-community.";
              type boolean;
            }
            choice validation-time {
              description
                "Sets prefix validation time (in seconds) or to set off the auto
matic prefix validation after an RPKI update.";
              case validation-off {
                leaf disable {
                  type boolean;
                }
              }
              case set-time {
                leaf prefix-validation-time {
                  description
                    "Range in seconds.";
                  type uint16 {
                    range "5..60";
                  }
                }
              }
            }
          }
          container bestpath-computation {
            description
              "Configures RPKI bestpath computation options.";
            leaf enable {
              description
                "Enables the validity states of BGP paths to affect the path's p
reference in the BGP bestpath process.";
              type boolean;
            }
            leaf allow-invalid {
              description
                "Allows all 'invalid' paths to be considered for BGP bestpath co
mputation.";
              type boolean;
            }
          }
        }
        uses bgp-neighbor-timers;
        container af-configuration {
          description
            "Top level container for address families specific configuration of
the BGP router.";
          container ipv4 {
            container mdt {
```

```
              container bgp {
                description
                  "BGP specific commands for ipv4-mdt address family/sub-addess
family combination.";
                uses bgp-dampening;
                leaf scan-time {
                  description
                    "Configure background scanner interval in seconds.";
                  type uint8 {
                    range "5..60";
                  }
                }
                uses slow-peer-config;
                leaf soft-reconfig-backup {
                  description
                    "Use soft-reconfiguration inbound only when route-refresh is
 not negotiated.";
                  type boolean;
                }
                leaf propagate-dmzlink-bw {
                  description
                    "Use DMZ Link Bandwidth as weight for BGP multipaths.";
                  type boolean;
                }
              }
            }
            container multicast {
              container bgp {
                description
                  "BGP specific commands for ipv4-multicast address family/sub-a
ddess family combination.";
                uses bgp-af-config;
              }
              leaf auto-summary {
                description
                  "Enable automatic network number summarization";
                type boolean;
              }
              uses router-af-config;
              leaf default-metric {
                description
                  "Set metric of redistributed routes.";
                type uint32;
              }
            }
            container unicast {
              container bgp {
                description
                  "BGP specific commands for ipv4-unicast address family/sub-add
ess family combination.";
                uses bgp-af-config;
                leaf always-compare-med {
```

```
                  description
                    "Allow comparing MED from different neighbors.";
                  type boolean;
                  default "false";
                }
                leaf enforce-first-as {
                  description
                    "Enforce the first AS for EBGP routes(default).";
                  type boolean;
                  default "true";
                }
                leaf fast-external-fallover {
                  description
                    "Immediately reset session if a link to a directly connected
 external peer goes down.";
                  type boolean;
                  default "true";
                }
                leaf suppress-inactive {
                  description
                    "Suppress routes that are not in the routing table.";
                  type boolean;
                }
                leaf asnotation {
                  description
                    "Sets the default asplain notation.";
                  type enumeration {
                    enum "asplain";
                    enum "dot";
                  }
                }
                leaf enable-client-to-client-reflection {
                  description
                    "Manages client to client route reflection.";
                  type boolean;
                  default "true";
                }
                leaf cluster-id {
                  description
                    "Configure Route-Reflector Cluster-id.";
                  type string;
                }
                container confederation {
                  description
                    "AS confederation parameters.";
                  leaf identifier {
                    description
                      "Confederation identifier.";
                    type string;
```

```
                  }
                  list peers {
                    description
                      "Confederation peers.";
                    key "as-name";
                    leaf as-name {
                      type string;
                    }
                  }
                }
                container consistency-checker {
                  description
                    "Consistency-checker configuration.";
                  leaf enable {
                    type boolean;
                  }
                  leaf interval {
                    description
                      "Check interval in minutes.";
                    type uint16 {
                      range "5..1440";
                    }
                  }
                  choice inconsistency-action {
                    case error-message {
                      description
                        "Specifies that when an inconsistency is found, the syst
em will only generate a syslog message.";
                      leaf generate-error-message-only {
                        type boolean;
                      }
                    }
                    case autorepair {
                      description
                        "Specifies that when an inconsistency is found,
                         the system will generate a syslog message and take acti
on
                         based on the type of inconsistency found.";
                      leaf perform-autorepair {
                        type boolean;
                      }
                    }
                  }
                }
                leaf deterministic-med {
                  description
                    "If enabled it enforce the deterministic comparison of the M
ED value between
                     all paths received from within the same autonomous system."
;
                  type boolean;
                }
```

```
                container graceful-restart {
                  description
                    "Controls the BGP graceful restart capability.";
                  leaf enable {
                    type boolean;
                  }
                  leaf restart-time {
                    description
                      "Sets the maximum time period (in seconds) that the local
router will wait
                       for a graceful-restart-capable neighbor to return to norm
al operation after a restart event occurs.";
                    type uint16 {
                      range "1..3600";
                    }
                    default "120";
                  }
                  leaf stalepath-time {
                    description
                      "Sets the maximum time period that the local router will h
old stale paths for a restarting peer.";
                    type uint16 {
                      range "5..3600";
                    }
                    default "360";
                  }
                }
                container listener-congfig {
                  description
                    "Associates a subnet range with a BGP peer group and activat
e the BGP dynamic neighbors feature.";
                  leaf enable {
                    type boolean;
                  }
                  leaf limit {
                    description
                      "Sets a maximum limit number of BGP dynamic subnet range n
eighbors.";
                    type uint16 {
                      range "1..5000";
                    }
                    default "100";
                  }
                  leaf range {
                    description
                      "Specifies a subnet range that is to be associated with a
specified peer group.";
                    type uint16 {
                      range "0..32";
                    }
                  }
                  leaf peer-group {
                    description
                      "Specifies a BGP peer group that is to be associated with
the specified subnet range.";
```

```
                    type string;
                  }
                }
                leaf log-neighbor-changes {
                  description
                    "Log neighbor up/down and reset reason.";
                  type boolean;
                }
                leaf max-as-limit {
                  description
                    "Configures BGP to discard routes that have a number of auto
nomous system numbers in AS-path that exceed the specified value.";
                  type uint16 {
                    range "1..254";
                  }
                }
                container transport {
                  description
                    "Manages transport session parameters.";
                  leaf enable-path-mtu-discovery {
                    description
                      "Enables transport path MTU discovery.";
                    type boolean;
                    default "true";
                  }
                }
              }
              leaf auto-summary {
                description
                  "Enable automatic network number summarization";
                type boolean;
              }
              uses router-af-config;
              uses maximum-paths;
              leaf synchronization {
                description
                  "Perform IGP synchronization.";
                type boolean;
              }
            }
            container mvpn {
              container bgp {
                description
                  "BGP specific commands for ipv4-mvpn address family/sub-addess
 family combination.";
                uses bgp-af-mvpn-config;
              }
              leaf auto-summary {
                description
                  "Enable automatic network number summarization.";
```

```
                type boolean;
              }
            }
          }
          container ipv6 {
            container multicast {
              container bgp {
                description
                  "BGP specific commands for ipv6-multicast address family/sub-a
ddess family combination.";
                uses bgp-af-config;
              }
              uses router-af-config;
            }
            container unicast {
              container bgp {
                description
                  "BGP specific commands for ipv6-unicast address family/sub-add
ess family combination.";
                uses bgp-af-config;
              }
              uses router-af-config;
              leaf default-metric {
                description
                  "Set metric of redistributed routes.";
                type uint32;
              }
              uses maximum-paths;
              leaf synchronization {
                description
                  "Perform IGP synchronization.";
                type boolean;
              }
            }
            container mvpn {
              container bgp {
                description
                  "BGP specific commands for ipv6-mvpn address family/sub-addess
 family combination.";
                uses bgp-af-mvpn-config;
              }
            }
          }
          container l2vpn {
            container vpls {
              container bgp {
                description
                  "BGP specific commands for l2vpn-vpls address family/sub-addes
s family combination.";
                leaf scan-time {
                  description
                    "Configure background scanner interval in seconds.";
```

```
                  type uint8 {
                    range "5..60";
                  }
                }
                uses slow-peer-config;
              }
            }
          }
          container nsap {
            container unicast {
              container bgp {
                description
                  "BGP specific commands for nsap-unicast address family/sub-add
ess family combination.";
                container aggregate-timer {
                  description
                    "Configure Aggregation Timer.";
                  leaf enable {
                    type boolean;
                    default "true";
                  }
                  leaf threshold {
                    type uint16 {
                      range "6..60";
                    }
                  }
                }
                uses bgp-dampening;
                leaf propagate-dmzlink-bw {
                  description
                    "Use DMZ Link Bandwidth as weight for BGP multipaths.";
                  type boolean;
                }
                leaf redistribute-internal {
                  description
                    "Allow redistribution of iBGP into IGPs (dangerous)";
                  type boolean;
                }
                leaf scan-time {
                  description
                    "Configure background scanner interval in seconds.";
                  type uint8 {
                    range "5..60";
                  }
                }
                uses slow-peer-config;
                leaf soft-reconfig-backup {
                  description
                    "Use soft-reconfiguration inbound only when route-refresh is
 not negotiated.";
```

```
                type boolean;
              }
            }
            leaf default-metric {
              description
                "Set metric of redistributed routes.";
              type uint32;
            }
            uses maximum-paths;
            leaf network {
              description
                "Specify a network to announce via BGP.";
              type inet:ip-address;
            }
            uses redistribute;
            leaf synchronization {
              description
                "Perform IGP synchronization.";
              type boolean;
            }
          }
        }
        container rtfilter {
          container unicast {
            container bgp {
              description
                "BGP specific commands for rtfilter-unicast address family/sub
-addess family combination.";
              uses slow-peer-config;
            }
            uses maximum-paths;
          }
        }
        container vpnv4 {
          container unicast {
            container bgp {
              description
                "BGP specific commands for vpnv4-unicast address family/sub-ad
dess family combination.";
              uses bgp-af-vpn-config;
            }
            uses maximum-paths;
          }
          container multicast {
            container bgp {
              description
                "BGP specific commands for vpnv4-multicast address family/sub-
addess family combination.";
              uses bgp-af-vpn-config;
            }
            uses maximum-paths;
```

```
            }
          }
          container vpnv6 {
            container unicast {
              container bgp {
                description
                  "BGP specific commands for vpnv6-unicast address family/sub-ad
dess family combination.";
                uses bgp-af-vpn-config;
              }
            }
          }
        }
      }
      container bgp-neighbors {
        description
          "The top level container for the list of neighbours of the BGP router.
";
        list bgp-neighbor {
          key "peer-address";
          leaf peer-address {
            type inet:ip-address;
            mandatory true;
          }
          uses bgp-neighbor-config;
          uses bgp-neighbor-transport-config;
          uses bgp-neighbor-timers;

          container bgp-neighbor-state {
            description
              "The operational parameters describing the neighbour state.
               It is intended that this container may be augmented by vendors to
 reflect the vendor-specific operational state parameters.";
            leaf adminStatus {
              type bgp-peer-admin-status;
            }
            leaf in-lastupdatetime {
              type yang:timestamp;
            }
          }
          container bgp-neighbor-statistics {
            description
              "The operational parameters describing the neighbour statistics.
               It is intended that this container may be augmented by vendors to
 reflect the vendor-specific staistical parameters.";
            leaf nr-in-updates {
              type uint32;
            }
            leaf nr-out-updates {
              type uint32;
            }
          }
```

```
        }
      }
      container bgp-neighbor-groups {
        description
          "The top level container for the list of neighbour groups of the BGP r
outer.";
        list bgp-neighbor-group {
            key "nbr-grp-name";
            leaf nbr-grp-name {
              type string;
              mandatory true;
            }
            uses bgp-neighbor-config;
            uses bgp-neighbor-timers;
        }
      }
      container prefix-lists {
        description
          "Contains all prefix lists defined
            on a router.";
        list prefix-list {
          key "prefix-list-name";
          description
            "A prefix list.";
          leaf prefix-list-name {
            type string;
          }
          container prefixes {
            list prefix {
              key "seq-nr";
              description
                "A prefix is a rule with a BGP filter.
                 The left hand side of the rule is the prefix filter.
                 It specifies a set of IP addresses.
                 If a BGP announcement contains an address that matches, the
                 rule is applied.  The right hand side of the rule specifies
                 the action that is to be applied.";
              leaf seq-nr {
                type uint16;
                description
                  "Sequence number of the rule.
                   The sequence number is included for compatibility purposes
                   with CLI; from a machine-to-machine interface perspective,
                   it would strictly speaking not be required as list elements
                   can be arranged in a particular order.";
              }
              container prefix-filter {
                choice ip-address-group {
                  case ip-address {
```

```
                    leaf ip-address {
                      type inet:ip-address;
                      mandatory true;
                    }
                  }
                  case prefix {
                    leaf prefix {
                      type inet:ip-prefix;
                      mandatory true;
                    }
                  }
                  case host {
                    leaf ip-host-address {
                      type inet:host;
                      mandatory true;
                    }
                  }
                  case ip-range {
                    leaf lower {
                      type inet:ip-address;
                    }
                    leaf upper {
                      type inet:ip-address;
                    }
                  }
                }
                leaf action {
                  type actions-enum;
                  mandatory true;
                  description
                    "permit/deny action";
                }
                container statistics {
                  leaf prefix-hit-count {
                    type uint32;
                    config false;
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}

</CODE ENDS>
```

5. IANA Considerations

6. Security Considerations

The transport protocol used for sending the BGP data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

7. Acknowledgements

The authors would like to thank Adam Simpson and Gunter Vandevelde for their detail reviews and comments.

8. Contributors

In addition to the authors listed on the front page, the following individuals have also helped to shape this document:

Dhanendra Jain

9. References

9.1. Normative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-15 (work in progress), May 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC2842] Chandra, R. and J. Scudder, "Capabilities Advertisement with BGP-4", RFC 2842, May 2000.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

9.2. Informative References

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, February 2009.

Authors' Addresses

Aleksandr Zhdankin
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: azhdanki@cisco.com

Keyur Patel
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

Alexander Clemm
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: alex@cisco.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Mahesh Jethanandani
Ciena
1741 Technology Drive
San Jose, CA 95110
USA

Email: mjethanandani@gmail.com

Xyfeng Liu
Ericsson
1595 Spring Hill Road, Suite 500
Vienna, VA 22182
USA

Email: xufeng.liu@ericsson.com