Network Working Group
INTERNET DRAFT

Y. Rekhter
Juniper Networks
T. Li
Procket Networks, Inc.
S. Hares
NextHop Technologies, Inc.
Editors

A Border Gateway Protocol 4 (BGP-4)
<draft-ietf-idr-bgp4-20.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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/ietf/1id-abstracts.txt

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

Specification of Requirements

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 [RFC2119].

Table of Contents

Abstract

The Border Gateway Protocol (BGP) is an inter-Autonomous System routing protocol.

The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of Autonomous Systems (ASs) that reachability information traverses. This information is sufficient to construct a graph of AS connectivity from which routing loops may be pruned and some policy decisions at the AS level may be enforced.

BGP-4 provides a set of mechanisms for supporting Classless Inter-Domain Routing (CIDR) [RFC1518, RFC1519]. These mechanisms include support for advertising a set of destinations as an IP prefix and eliminating the concept of network "class" within BGP. BGP-4 also introduces mechanisms which allow aggregation of routes, including aggregation of AS paths.

Routing information exchanged via BGP supports only the destination-based forwarding paradigm, which assumes that a router forwards a packet based solely on the destination address carried in the IP header of the packet. This, in turn, reflects the set of policy decisions that can (and can not) be enforced using BGP. BGP can support only the policies conforming to the destination-based forwarding paradigm.

1. Definition of commonly used terms

This section provides definition for terms that have a specific meaning to the BGP protocol and that are used throughout the text.

Adj-RIB-In
: The Adj-RIBs-In contain unprocessed routing information that has been advertised to the local BGP speaker by its peers.

Adj-RIB-Out
: The Adj-RIBs-Out contains the routes for advertisement to specific peers by means of the local speaker's UPDATE messages.

Autonomous System (AS)
: The classic definition of an Autonomous System is a set of routers under a single technical administration, using an interior gateway protocol (IGP) and common metrics to determine how to route packets within the AS, and using an inter-AS routing protocol to determine how to route packets to other ASs. Since this classic

definition was developed, it has become common for a single AS to use several IGPs and sometimes several sets of metrics within an AS. The use of the term Autonomous System here stresses the fact that, even when multiple IGPs and metrics are used, the administration of an AS appears to other ASs to have a single coherent interior routing plan and presents a consistent picture of what destinations are reachable through it.

BGP Identifier
A 4-octet unsigned integer indicating the BGP Identifier of the sender of BGP messages. A given BGP speaker sets the value of its BGP Identifier to an IP address assigned to that BGP speaker. The value of the BGP Identifier is determined on startup and is the same for every local interface and every BGP peer.

BGP speaker
A router that implements BGP.

EBGP
External BGP (BGP connection between external peers).

External peer
Peer that is in a different Autonomous System than the local system.

Feasible route
A route that is available for use.

IBGP
Internal BGP (BGP connection between internal peers).

Internal peer
Peer that is in the same Autonomous System as the local system.

IGP
Interior Gateway Protocol - a routing protocol used to exchange routing information among routers within a single Autonomous System.

Loc-RIB
The Loc-RIB contains the routes that have been selected by the local BGP speaker's Decision Process.

NLRI
Network Layer Reachability Information.

Route
A unit of information that pairs a set of destinations with the

attributes of a path to those destinations. The set of destinations are systems whose IP addresses are contained in one IP address prefix carried in the Network Layer Reachability Information (NLRI) field of an UPDATE message. The path is the information reported in the path attributes field of the same UPDATE message.

RIB
Routing Information Base.

Unfeasible route
A previously advertised feasible route that is no longer available for use.

2. Acknowledgments

This document was originally published as RFC 1267 in October 1991, jointly authored by Kirk Lougheed and Yakov Rekhter.

We would like to express our thanks to Guy Almes, Len Bosack, and Jeffrey C. Honig for their contributions to the earlier version (BGP-1) of this document.

We would like to specially acknowledge numerous contributions by Dennis Ferguson to the earlier version of this document.

We like to explicitly thank Bob Braden for the review of the earlier version (BGP-2) of this document as well as his constructive and valuable comments.

We would also like to thank Bob Hinden, Director for Routing of the Internet Engineering Steering Group, and the team of reviewers he assembled to review the earlier version (BGP-2) of this document. This team, consisting of Deborah Estrin, Milo Medin, John Moy, Radia Perlman, Martha Steenstrup, Mike St. Johns, and Paul Tsuchiya, acted with a strong combination of toughness, professionalism, and courtesy.

Certain sections of the document borrowed heavily from IDRP [IS10747], which is the OSI counterpart of BGP. For this credit should be given to the ANSI X3S3.3 group chaired by Lyman Chapin and to Charles Kunzinger who was the IDRP editor within that group.

We would also like to thank Benjamin Abarbanel, Enke Chen, Edward Crabbe, Mike Craren, Vincent Gillet, Eric Gray, Jeffrey Haas, Dimitry Haskin, John Krawczyk, David LeRoy, Dan Massey, Jonathan Natale, Dan Pei, Mathew Richardson, John Scudder, John Stewart III, Dave Thaler,

Paul Traina, Russ White, Curtis Villamizar, and Alex Zinin for their comments.

We would like to specially acknowledge Andrew Lange for his help in preparing the final version of this document.

Finally, we would like to thank all the members of the IDR Working Group for their ideas and support they have given to this document.

3. Summary of Operation

The Border Gateway Protocol (BGP) is an inter-Autonomous System routing protocol. It is built on experience gained with EGP as defined in [RFC904] and EGP usage in the NSFNET Backbone as described in [RFC1092] and [RFC1093].

The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of Autonomous Systems (ASs) that reachability information traverses. This information is sufficient to construct a graph of AS connectivity from which routing loops may be pruned and some policy decisions at the AS level may be enforced.

In the context of this document we assume that a BGP speaker advertises to its peers only those routes that it itself uses (in this context a BGP speaker is said to "use" a BGP route if it is the most preferred BGP route and is used in forwarding). All other cases are outside the scope of this document.

In the context of this document the term "IP address" refers to an IP Version 4 address [RFC791].

Routing information exchanged via BGP supports only the destination-based forwarding paradigm, which assumes that a router forwards a packet based solely on the destination address carried in the IP header of the packet. This, in turn, reflects the set of policy decisions that can (and can not) be enforced using BGP. Note that some policies can not be supported by the destination-based forwarding paradigm, and thus require techniques such as source routing (aka explicit routing) to be enforced. Such policies can not be enforced using BGP either. For example, BGP does not enable one AS to send traffic to a neighboring AS for forwarding to some destination (reachable through but) beyond that neighboring AS intending that the traffic take a different route to that taken by the traffic originating in the neighboring AS (for that same destination). On the other hand, BGP can support any policy conforming to the destination-based

forwarding paradigm.

BGP-4 provides a new set of mechanisms for supporting Classless Inter-Domain Routing (CIDR) [RFC1518, RFC1519]. These mechanisms include support for advertising a set of destinations as an IP prefix and eliminating the concept of network "class" within BGP. BGP-4 also introduces mechanisms which allow aggregation of routes, including aggregation of AS paths.

This document uses the term `Autonomous System' (AS) throughout. The classic definition of an Autonomous System is a set of routers under a single technical administration, using an interior gateway protocol (IGP) and common metrics to determine how to route packets within the AS, and using an inter-AS routing protocol to determine how to route packets to other ASs. Since this classic definition was developed, it has become common for a single AS to use several IGPs and sometimes several sets of metrics within an AS. The use of the term Autonomous System here stresses the fact that, even when multiple IGPs and metrics are used, the administration of an AS appears to other ASs to have a single coherent interior routing plan and presents a consistent picture of what destinations are reachable through it.

BGP uses TCP [RFC793] as its transport protocol. This eliminates the need to implement explicit update fragmentation, retransmission, acknowledgment, and sequencing. BGP listens on TCP port 179. The error notification mechanism used in BGP assumes that TCP supports a "graceful" close, i.e., that all outstanding data will be delivered before the connection is closed.

Two systems form a TCP connection between one another. They exchange messages to open and confirm the connection parameters.

The initial data flow is the portion of the BGP routing table that is allowed by the export policy, called the Adj-Ribs-Out (see 3.2). Incremental updates are sent as the routing tables change. BGP does not require periodic refresh of the routing table. To allow local policy changes to have the correct effect without resetting any BGP connections, a BGP speaker SHOULD either (a) retain the current version of the routes advertised to it by all of its peers for the duration of the connection, or (b) make use of the Route Refresh extension [RFC2918].

KEEPALIVE messages may be sent periodically to ensure the liveness of the connection. NOTIFICATION messages are sent in response to errors or special conditions. If a connection encounters an error condition, a NOTIFICATION message is sent and the connection is closed.

A peer in a different AS is referred to as an external peer, while a

peer in the same AS is referred to as an internal peer. Internal BGP and external BGP are commonly abbreviated IBGP and EBGP.

If a particular AS has multiple BGP speakers and is providing transit service for other ASs, then care must be taken to ensure a consistent view of routing within the AS. A consistent view of the interior routes of the AS is provided by the IGP used within the AS. For the purpose of this document, it is assumed that a consistent view of the routes exterior to the AS is provided by having all BGP speakers within the AS maintain IBGP with each other. Care must be taken to ensure that the interior routers have all been updated with transit information before the BGP speakers announce to other ASs that transit service is being provided.

This document specifies the base behavior of the BGP protocol. This behavior can and is modified by extention specifications. When the protocol is extended the new behavior is fully documented in the extention specifications.

3.1 Routes: Advertisement and Storage

For the purpose of this protocol, a route is defined as a unit of information that pairs a set of destinations with the attributes of a path to those destinations. The set of destinations are systems whose IP addresses are contained in one IP address prefix carried in the Network Layer Reachability Information (NLRI) field of an UPDATE message, and the path is the information reported in the path attributes field of the same UPDATE message.

Routes are advertised between BGP speakers in UPDATE messages. Multiple routes that have the same path attributes can be advertised in a single UPDATE message by including multiple prefixes in the NLRI field of the UPDATE message.

Routes are stored in the Routing Information Bases (RIBs): namely, the Adj-RIBs-In, the Loc-RIB, and the Adj-RIBs-Out, as described in Section 3.2.

If a BGP speaker chooses to advertise the route, it MAY add to or modify the path attributes of the route before advertising it to a peer.

BGP provides mechanisms by which a BGP speaker can inform its peer that a previously advertised route is no longer available for use. There are three methods by which a given BGP speaker can indicate that a route has been withdrawn from service:

a) the IP prefix that expresses the destination for a previously advertised route can be advertised in the WITHDRAWN ROUTES field in the UPDATE message, thus marking the associated route as being no longer available for use

b) a replacement route with the same NLRI can be advertised, or

c) the BGP speaker - BGP speaker connection can be closed, which implicitly removes from service all routes which the pair of speakers had advertised to each other.

Changing attribute of a route is accomplished by advertising a replacement route. The replacement route carries new (changed) attributes and has the same NLRI as the original route.

3.2 Routing Information Bases

The Routing Information Base (RIB) within a BGP speaker consists of three distinct parts:

a) Adj-RIBs-In: The Adj-RIBs-In store routing information that has been learned from inbound UPDATE messages received from other BGP speakers. Their contents represent routes that are available as an input to the Decision Process.

b) Loc-RIB: The Loc-RIB contains the local routing information that the BGP speaker has selected by applying its local policies to the routing information contained in its Adj-RIBs-In. These are the routes that will be used by the local BGP speaker. The next hop for each of these routes MUST be resolvable via the local BGP speaker's Routing Table.

c) Adj-RIBs-Out: The Adj-RIBs-Out store the information that the local BGP speaker has selected for advertisement to its peers. The routing information stored in the Adj-RIBs-Out will be carried in the local BGP speaker's UPDATE messages and advertised to its peers.

In summary, the Adj-RIBs-In contain unprocessed routing information that has been advertised to the local BGP speaker by its peers; the Loc-RIB contains the routes that have been selected by the local BGP speaker's Decision Process; and the Adj-RIBs-Out organize the routes for advertisement to specific peers by means of the local speaker's UPDATE messages.

Although the conceptual model distinguishes between Adj-RIBs-In, Loc-RIB, and Adj-RIBs-Out, this neither implies nor requires that an

implementation must maintain three separate copies of the routing information. The choice of implementation (for example, 3 copies of the information vs 1 copy with pointers) is not constrained by the protocol.

Routing information that the BGP speaker uses to forward packets (or to construct the forwarding table that is used for packet forwarding) is maintained in the Routing Table. The Routing Table accumulates routes to directly connected networks, static routes, routes learned from the IGP protocols, and routes learned from BGP. Whether or not a specific BGP route should be installed in the Routing Table, and whether a BGP route should override a route to the same destination installed by another source is a local policy decision, not specified in this document. Besides actual packet forwarding, the Routing Table is used for resolution of the next-hop addresses specified in BGP updates (see Section 5.1.3).

4. Message Formats

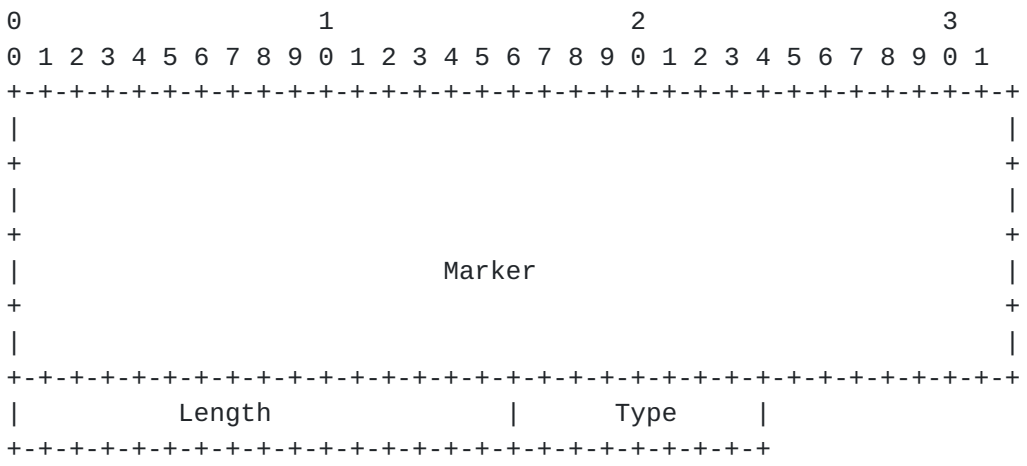
This section describes message formats used by BGP.

BGP messages are sent over a TCP connection. A message is processed only after it is entirely received. The maximum message size is 4096 octets. All implementations are required to support this maximum message size. The smallest message that may be sent consists of a BGP header without a data portion, or 19 octets.

All multi-octet fields are in network byte order.

4.1 Message Header Format

Each message has a fixed-size header. There may or may not be a data portion following the header, depending on the message type. The layout of these fields is shown below:



Marker:

This 16-octet field is included for compatibility; it MUST be set to all ones.

Length:

This 2-octet unsigned integer indicates the total length of the message, including the header, in octets. Thus, e.g., it allows one to locate in the TCP stream the (Marker field of the) next message. The value of the Length field MUST always be at least 19 and no greater than 4096, and MAY be further constrained, depending on the message type. No "padding" of extra data after the message is allowed, so the Length field MUST have the smallest value required given the rest of the message.

Type:

This 1-octet unsigned integer indicates the type code of the message. This document defines the following type codes:

1 - OPEN
2 - UPDATE
3 - NOTIFICATION
4 - KEEPALIVE

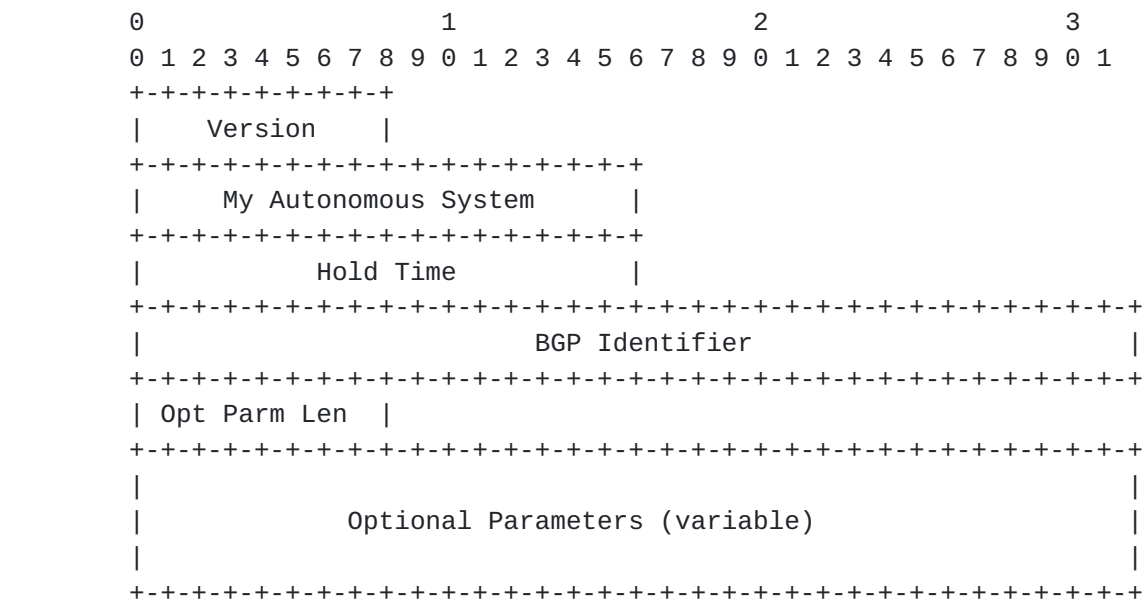
[RFC2918] defines one more type code.

4.2 OPEN Message Format

After a TCP is established, the first message sent by each side is an OPEN message. If the OPEN message is acceptable, a KEEPALIVE message

confirming the OPEN is sent back.

In addition to the fixed-size BGP header, the OPEN message contains the following fields:



Version:

This 1-octet unsigned integer indicates the protocol version number of the message. The current BGP version number is 4.

My Autonomous System:

This 2-octet unsigned integer indicates the Autonomous System number of the sender.

Hold Time:

This 2-octet unsigned integer indicates the number of seconds that the sender proposes for the value of the Hold Timer. Upon receipt of an OPEN message, a BGP speaker MUST calculate the value of the Hold Timer by using the smaller of its configured Hold Time and the Hold Time received in the OPEN message. The Hold Time MUST be either zero or at least three seconds. An implementation MAY reject connections on the basis of the Hold Time. The calculated value indicates the maximum number of seconds that may elapse between the receipt of successive KEEPALIVE, and/or UPDATE messages by the sender.

BGP Identifier:

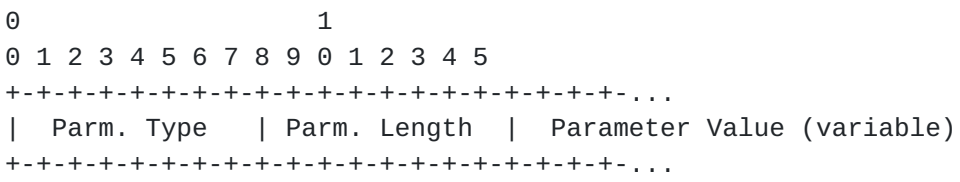
This 4-octet unsigned integer indicates the BGP Identifier of the sender. A given BGP speaker sets the value of its BGP Identifier to an IP address assigned to that BGP speaker. The value of the BGP Identifier is determined on startup and is the same for every local interface and every BGP peer.

Optional Parameters Length:

This 1-octet unsigned integer indicates the total length of the Optional Parameters field in octets. If the value of this field is zero, no Optional Parameters are present.

Optional Parameters:

This field contains a list of optional parameters, where each parameter is encoded as a <Parameter Type, Parameter Length, Parameter Value> triplet.



Parameter Type is a one octet field that unambiguously identifies individual parameters. Parameter Length is a one octet field that contains the length of the Parameter Value field in octets. Parameter Value is a variable length field that is interpreted according to the value of the Parameter Type field.

[RFC2842] defines the Capabilities Optional Parameter.

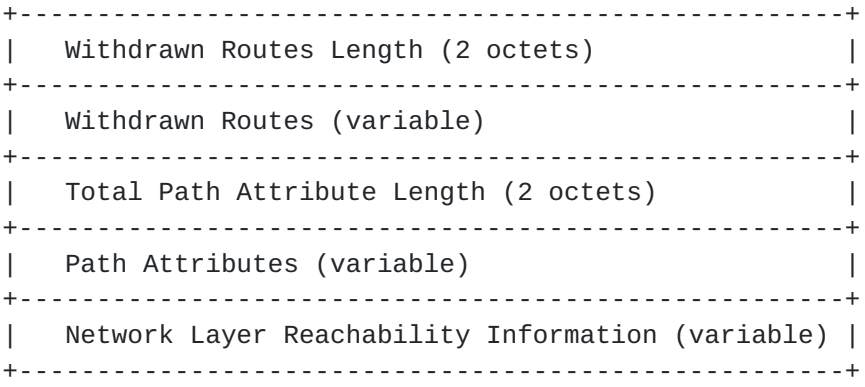
The minimum length of the OPEN message is 29 octets (including message header).

4.3 UPDATE Message Format

UPDATE messages are used to transfer routing information between BGP peers. The information in the UPDATE message can be used to construct a graph describing the relationships of the various Autonomous Systems. By applying rules to be discussed, routing information loops and some other anomalies may be detected and removed from inter-AS routing.

An UPDATE message is used to advertise feasible routes sharing common path attributes to a peer, or to withdraw multiple unfeasible routes

from service (see 3.1). An UPDATE message MAY simultaneously advertise a feasible route and withdraw multiple unfeasible routes from service. The UPDATE message always includes the fixed-size BGP header, and also includes the other fields as shown below (note, some of the shown fields may not be present in every UPDATE message):



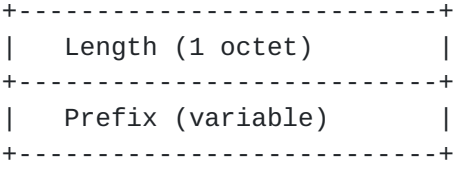
Withdrawn Routes Length:

This 2-octets unsigned integer indicates the total length of the Withdrawn Routes field in octets. Its value allows the length of the Network Layer Reachability Information field to be determined as specified below.

A value of 0 indicates that no routes are being withdrawn from service, and that the WITHDRAWN ROUTES field is not present in this UPDATE message.

Withdrawn Routes:

This is a variable length field that contains a list of IP address prefixes for the routes that are being withdrawn from service. Each IP address prefix is encoded as a 2-tuple of the form <length, prefix>, whose fields are described below:



The use and the meaning of these fields are as follows:

a) Length:

The Length field indicates the length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with prefix, itself, of zero octets).

b) Prefix:

The Prefix field contains an IP address prefix followed by the minimum number of trailing bits needed to make the end of the field fall on an octet boundary. Note that the value of trailing bits is irrelevant.

Total Path Attribute Length:

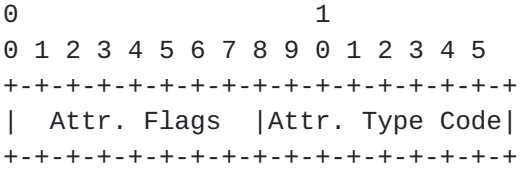
This 2-octet unsigned integer indicates the total length of the Path Attributes field in octets. Its value allows the length of the Network Layer Reachability field to be determined as specified below.

A value of 0 indicates that no Network Layer Reachability Information field is present in this UPDATE message.

Path Attributes:

A variable length sequence of path attributes is present in every UPDATE message, except for an UPDATE message that carries only the withdrawn routes. Each path attribute is a triple <attribute type, attribute length, attribute value> of variable length.

Attribute Type is a two-octet field that consists of the Attribute Flags octet followed by the Attribute Type Code octet.



The high-order bit (bit 0) of the Attribute Flags octet is the Optional bit. It defines whether the attribute is optional (if set to 1) or well-known (if set to 0).

The second high-order bit (bit 1) of the Attribute Flags octet is the Transitive bit. It defines whether an optional attribute is transitive (if set to 1) or non-transitive (if set to 0). For well-known attributes, the Transitive bit MUST be set to 1.

(See Section 5 for a discussion of transitive attributes.)

The third high-order bit (bit 2) of the Attribute Flags octet is the Partial bit. It defines whether the information contained in the optional transitive attribute is partial (if set to 1) or complete (if set to 0). For well-known attributes and for optional non-transitive attributes the Partial bit MUST be set to 0.

The fourth high-order bit (bit 3) of the Attribute Flags octet is the Extended Length bit. It defines whether the Attribute Length is one octet (if set to 0) or two octets (if set to 1).

The lower-order four bits of the Attribute Flags octet are unused. They MUST be zero when sent and MUST be ignored when received.

The Attribute Type Code octet contains the Attribute Type Code. Currently defined Attribute Type Codes are discussed in Section 5.

If the Extended Length bit of the Attribute Flags octet is set to 0, the third octet of the Path Attribute contains the length of the attribute data in octets.

If the Extended Length bit of the Attribute Flags octet is set to 1, then the third and the fourth octets of the path attribute contain the length of the attribute data in octets.

The remaining octets of the Path Attribute represent the attribute value and are interpreted according to the Attribute Flags and the Attribute Type Code. The supported Attribute Type Codes, their attribute values and uses are the following:

a) ORIGIN (Type Code 1):

ORIGIN is a well-known mandatory attribute that defines the origin of the path information. The data octet can assume the following values:

Value	Meaning
0	IGP - Network Layer Reachability Information is interior to the originating AS
1	EGP - Network Layer Reachability Information learned via the EGP protocol [RFC904]

2	INCOMPLETE - Network Layer Reachability Information learned by some other means

Usage of this attribute is defined in 5.1.1.

b) AS_PATH (Type Code 2):

AS_PATH is a well-known mandatory attribute that is composed of a sequence of AS path segments. Each AS path segment is represented by a triple <path segment type, path segment length, path segment value>.

The path segment type is a 1-octet long field with the following values defined:

Value	Segment Type
1	AS_SET: unordered set of ASs a route in the UPDATE message has traversed
2	AS_SEQUENCE: ordered set of ASs a route in the UPDATE message has traversed

The path segment length is a 1-octet long field containing the number of ASs (not the number of octets) in the path segment value field.

The path segment value field contains one or more AS numbers, each encoded as a 2-octets long field.

Usage of this attribute is defined in 5.1.2.

c) NEXT_HOP (Type Code 3):

This is a well-known mandatory attribute that defines the (unicast) IP address of the router that SHOULD be used as the next hop to the destinations listed in the Network Layer Reachability Information field of the UPDATE message.

Usage of this attribute is defined in 5.1.3.

d) MULTI_EXIT_DISC (Type Code 4):

This is an optional non-transitive attribute that is a four octet unsigned integer. The value of this attribute MAY be used by a BGP speaker's decision process to discriminate among multiple entry points to a neighboring autonomous

system.

Usage of this attribute is defined in 5.1.4.

e) LOCAL_PREF (Type Code 5):

LOCAL_PREF is a well-known attribute that is a four octet unsigned integer. A BGP speaker uses it to inform other internal peers of the advertising speaker's degree of preference for an advertised route.

Usage of this attribute is defined in 5.1.5.

f) ATOMIC_AGGREGATE (Type Code 6)

ATOMIC_AGGREGATE is a well-known discretionary attribute of length 0.

Usage of this attribute is defined in 5.1.6.

g) AGGREGATOR (Type Code 7)

AGGREGATOR is an optional transitive attribute of length 6. The attribute contains the last AS number that formed the aggregate route (encoded as 2 octets), followed by the IP address of the BGP speaker that formed the aggregate route (encoded as 4 octets). This SHOULD be the same address as the one used for the BGP Identifier of the speaker.

Usage of this attribute is defined in 5.1.7.

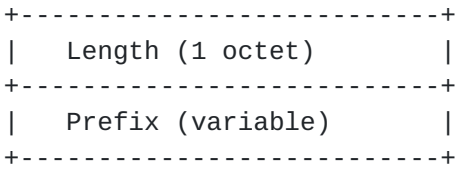
Network Layer Reachability Information:

This variable length field contains a list of IP address prefixes. The length in octets of the Network Layer Reachability Information is not encoded explicitly, but can be calculated as:

UPDATE message Length - 23 - Total Path Attributes Length - Withdrawn Routes Length

where UPDATE message Length is the value encoded in the fixed-size BGP header, Total Path Attribute Length and Withdrawn Routes Length are the values encoded in the variable part of the UPDATE message, and 23 is a combined length of the fixed-size BGP header, the Total Path Attribute Length field and the Withdrawn Routes Length field.

Reachability information is encoded as one or more 2-tuples of the form <length, prefix>, whose fields are described below:



The use and the meaning of these fields are as follows:

a) Length:

The Length field indicates the length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with prefix, itself, of zero octets).

b) Prefix:

The Prefix field contains an IP address prefix followed by enough trailing bits to make the end of the field fall on an octet boundary. Note that the value of the trailing bits is irrelevant.

The minimum length of the UPDATE message is 23 octets -- 19 octets for the fixed header + 2 octets for the Withdrawn Routes Length + 2 octets for the Total Path Attribute Length (the value of Withdrawn Routes Length is 0 and the value of Total Path Attribute Length is 0).

An UPDATE message can advertise at most one set of path attributes, but multiple destinations, provided that the destinations share these attributes. All path attributes contained in a given UPDATE message apply to all destinations carried in the NLRI field of the UPDATE message.

An UPDATE message can list multiple routes to be withdrawn from service. Each such route is identified by its destination (expressed as an IP prefix), which unambiguously identifies the route in the context of the BGP speaker - BGP speaker connection to which it has been previously advertised.

An UPDATE message might advertise only routes to be withdrawn from service, in which case it will not include path attributes or Network Layer Reachability Information. Conversely, it may advertise only a feasible route, in which case the WITHDRAWN ROUTES field need not be present.

An UPDATE message SHOULD NOT include the same address prefix in the WITHDRAWN ROUTES and Network Layer Reachability Information fields, however a BGP speaker MUST be able to process UPDATE messages in this form. A BGP speaker SHOULD treat an UPDATE message of this form as if the WITHDRAWN ROUTES doesn't contain the address prefix.

4.4 KEEPALIVE Message Format

BGP does not use any TCP-based keep-alive mechanism to determine if peers are reachable. Instead, KEEPALIVE messages are exchanged between peers often enough as not to cause the Hold Timer to expire. A reasonable maximum time between KEEPALIVE messages would be one third of the Hold Time interval. KEEPALIVE messages MUST NOT be sent more frequently than one per second. An implementation MAY adjust the rate at which it sends KEEPALIVE messages as a function of the Hold Time interval.

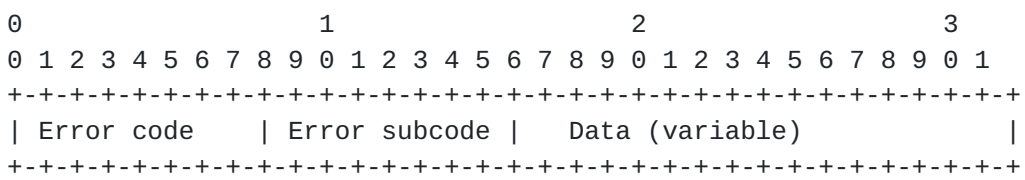
If the negotiated Hold Time interval is zero, then periodic KEEPALIVE messages MUST NOT be sent.

A KEEPALIVE message consists of only message header and has a length of 19 octets.

4.5 NOTIFICATION Message Format

A NOTIFICATION message is sent when an error condition is detected. The BGP connection is closed immediately after sending it.

In addition to the fixed-size BGP header, the NOTIFICATION message contains the following fields:



Error Code:

This 1-octet unsigned integer indicates the type of NOTIFICATION. The following Error Codes have been defined:

Error Code	Symbolic Name	Reference

1	Message Header Error	Section 6.1
2	OPEN Message Error	Section 6.2
3	UPDATE Message Error	Section 6.3
4	Hold Timer Expired	Section 6.5
5	Finite State Machine Error	Section 6.6
6	Cease	Section 6.7

Error subcode:

This 1-octet unsigned integer provides more specific information about the nature of the reported error. Each Error Code may have one or more Error Subcodes associated with it. If no appropriate Error Subcode is defined, then a zero (Unspecific) value is used for the Error Subcode field.

Message Header Error subcodes:

1 - Connection Not Synchronized.
2 - Bad Message Length.
3 - Bad Message Type.

OPEN Message Error subcodes:

1 - Unsupported Version Number.
2 - Bad Peer AS.
3 - Bad BGP Identifier.
4 - Unsupported Optional Parameter.
5 - [Deprecated - see Appendix A].
6 - Unacceptable Hold Time.

UPDATE Message Error subcodes:

1 - Malformed Attribute List.
2 - Unrecognized Well-known Attribute.
3 - Missing Well-known Attribute.
4 - Attribute Flags Error.
5 - Attribute Length Error.
6 - Invalid ORIGIN Attribute.
7 - [Deprecated - see Appendix A].
8 - Invalid NEXT_HOP Attribute.
9 - Optional Attribute Error.
10 - Invalid Network Field.

11 - Malformed AS_PATH.

Data:

This variable-length field is used to diagnose the reason for the NOTIFICATION. The contents of the Data field depend upon the Error Code and Error Subcode. See Section 6 below for more details.

Note that the length of the Data field can be determined from the message Length field by the formula:

Message Length = 21 + Data Length

The minimum length of the NOTIFICATION message is 21 octets (including message header).

5. Path Attributes

This section discusses the path attributes of the UPDATE message.

Path attributes fall into four separate categories:

1. Well-known mandatory.
2. Well-known discretionary.
3. Optional transitive.
4. Optional non-transitive.

Well-known attributes MUST be recognized by all BGP implementations. Some of these attributes are mandatory and MUST be included in every UPDATE message that contains NLRI. Others are discretionary and MAY or MAY NOT be sent in a particular UPDATE message.

All well-known attributes MUST be passed along (after proper updating, if necessary) to other BGP peers.

In addition to well-known attributes, each path MAY contain one or more optional attributes. It is not required or expected that all BGP implementations support all optional attributes. The handling of an unrecognized optional attribute is determined by the setting of the Transitive bit in the attribute flags octet. Paths with unrecognized transitive optional attributes SHOULD be accepted. If a path with unrecognized transitive optional attribute is accepted and passed along to other BGP peers, then the unrecognized transitive optional attribute of that path MUST be passed along with the path to other

BGP peers with the Partial bit in the Attribute Flags octet set to 1. If a path with recognized transitive optional attribute is accepted and passed along to other BGP peers and the Partial bit in the Attribute Flags octet is set to 1 by some previous AS, it is not set back to 0 by the current AS. Unrecognized non-transitive optional attributes MUST be quietly ignored and not passed along to other BGP peers.

New transitive optional attributes MAY be attached to the path by the originator or by any other BGP speaker in the path. If they are not attached by the originator, the Partial bit in the Attribute Flags octet is set to 1. The rules for attaching new non-transitive optional attributes will depend on the nature of the specific attribute. The documentation of each new non-transitive optional attribute will be expected to include such rules. (The description of the MULTI_EXIT_DISC attribute gives an example.) All optional attributes (both transitive and non-transitive) MAY be updated (if appropriate) by BGP speakers in the path.

The sender of an UPDATE message SHOULD order path attributes within the UPDATE message in ascending order of attribute type. The receiver of an UPDATE message MUST be prepared to handle path attributes within the UPDATE message that are out of order.

The same attribute (attribute with the same type) can not appear more than once within the Path Attributes field of a particular UPDATE message.

The mandatory category refers to an attribute which MUST be present in both IBGP and EBGP exchanges if NLRI are contained in the UPDATE message. Attributes classified as optional for the purpose of the protocol extension mechanism may be purely discretionary, or discretionary, required, or disallowed in certain contexts.

attribute	EBGP	IBGP
ORIGIN	mandatory	mandatory
AS_PATH	mandatory	mandatory
NEXT_HOP	mandatory	mandatory
MULTI_EXIT_DISC	discretionary	discretionary
LOCAL_PREF	see Section 5.1.5	required
ATOMIC_AGGREGATE	see Section 5.1.6 and 9.1.4	
AGGREGATOR	discretionary	discretionary

5.1 Path Attribute Usage

The usage of each BGP path attribute is described in the following clauses.

5.1.1 ORIGIN

ORIGIN is a well-known mandatory attribute. The ORIGIN attribute is generated by the speaker that originates the associated routing information. Its value SHOULD NOT be changed by any other speaker.

5.1.2 AS_PATH

AS_PATH is a well-known mandatory attribute. This attribute identifies the autonomous systems through which routing information carried in this UPDATE message has passed. The components of this list can be AS_SETs or AS_SEQUENCEs.

When a BGP speaker propagates a route which it has learned from another BGP speaker's UPDATE message, it modifies the route's AS_PATH attribute based on the location of the BGP speaker to which the route will be sent:

a) When a given BGP speaker advertises the route to an internal peer, the advertising speaker SHALL NOT modify the AS_PATH attribute associated with the route.

b) When a given BGP speaker advertises the route to an external peer, then the advertising speaker updates the AS_PATH attribute as follows:

1) if the first path segment of the AS_PATH is of type AS_SEQUENCE, the local system prepends its own AS number as the last element of the sequence (put it in the leftmost position). If the act of prepending will cause an overflow in the AS_PATH segment, i.e. more than 255 ASs, it is legal to prepend a new segment of type AS_SEQUENCE and prepend its own AS number to this new segment.

2) if the first path segment of the AS_PATH is of type AS_SET, the local system prepends a new path segment of type AS_SEQUENCE to the AS_PATH, including its own AS number in that

segment.

When a BGP speaker originates a route then:

a) the originating speaker includes its own AS number in a path segment of type AS_SEQUENCE in the AS_PATH attribute of all UPDATE messages sent to an external peer. (In this case, the AS number of the originating speaker's autonomous system will be the only entry the path segment, and this path segment will be the only segment in the AS_PATH attribute).

b) the originating speaker includes an empty AS_PATH attribute in all UPDATE messages sent to internal peers. (An empty AS_PATH attribute is one whose length field contains the value zero).

Whenever the modification of the AS_PATH attribute calls for including or prepending the AS number of the local system, the local system MAY include/prepend more than one instance of its own AS number in the AS_PATH attribute. This is controlled via local configuration.

5.1.3 NEXT_HOP

The NEXT_HOP is a well-known mandatory attribute that defines the IP address of the router that SHOULD be used as the next hop to the destinations listed in the UPDATE message. The NEXT_HOP attribute is calculated as follows.

1) When sending a message to an internal peer, if the route is not locally originated the BGP speaker SHOULD NOT modify the NEXT_HOP attribute, unless it has been explicitly configured to announce its own IP address as the NEXT_HOP. When announcing a locally originated route to an internal peer, the BGP speaker SHOULD use as the NEXT_HOP the interface address of the router through which the announced network is reachable for the speaker; if the route is directly connected to the speaker, or the interface address of the router through which the announced network is reachable for the speaker is the internal peer's address, then the BGP speaker SHOULD use for the NEXT_HOP attribute its own IP address (the address of the interface that is used to reach the peer).

2) When sending a message to an external peer X, and the peer is one IP hop away from the speaker:

- If the route being announced was learned from an internal peer or is locally originated, the BGP speaker can use for the NEXT_HOP attribute an interface address of the internal peer

router (or the internal router) through which the announced network is reachable for the speaker, provided that peer X shares a common subnet with this address. This is a form of "third party" NEXT_HOP attribute.

- Otherwise, if the route being announced was learned from an external peer, the speaker can use in the NEXT_HOP attribute an IP address of any adjacent router (known from the received NEXT_HOP attribute) that the speaker itself uses for local route calculation, provided that peer X shares a common subnet with this address. This is a second form of "third party" NEXT_HOP attribute.

- Otherwise, if the external peer to which the route is being advertised shares a common subnet with one of the interfaces of the announcing BGP speaker, the speaker MAY use the IP address associated with such an interface in the NEXT_HOP attribute. This is known as a "first party" NEXT_HOP attribute.

- By default (if none of the above conditions apply), the BGP speaker SHOULD use in the NEXT_HOP attribute the IP address of the interface that the speaker uses to establish the BGP connection to peer X.

3) When sending a message to an external peer X, and the peer is multiple IP hops away from the speaker (aka "multihop EBGP"):

- The speaker MAY be configured to propagate the NEXT_HOP attribute. In this case when advertising a route that the speaker learned from one of its peers, the NEXT_HOP attribute of the advertised route is exactly the same as the NEXT_HOP attribute of the learned route (the speaker just doesn't modify the NEXT_HOP attribute).

- By default, the BGP speaker SHOULD use in the NEXT_HOP attribute the IP address of the interface that the speaker uses to establish the BGP connection to peer X.

Normally the NEXT_HOP attribute is chosen such that the shortest available path will be taken. A BGP speaker MUST be able to support disabling advertisement of third party NEXT_HOP attributes to handle imperfectly bridged media.

A route originated by a BGP speaker SHALL NOT be advertised to a peer using an address of that peer as NEXT_HOP. A BGP speaker SHALL NOT install a route with itself as the next hop.

The NEXT_HOP attribute is used by the BGP speaker to determine the

actual outbound interface and immediate next-hop address that SHOULD be used to forward transit packets to the associated destinations.

The immediate next-hop address is determined by performing a recursive route lookup operation for the IP address in the NEXT_HOP attribute using the contents of the Routing Table, selecting one entry if multiple entries of equal cost exist. The Routing Table entry which resolves the IP address in the NEXT_HOP attribute will always specify the outbound interface. If the entry specifies an attached subnet, but does not specify a next-hop address, then the address in the NEXT_HOP attribute SHOULD be used as the immediate next-hop address. If the entry also specifies the next-hop address, this address SHOULD be used as the immediate next-hop address for packet forwarding.

5.1.4 MULTI_EXIT_DISC

The MULTI_EXIT_DISC is an optional non-transitive attribute which is intended to be used on external (inter-AS) links to discriminate among multiple exit or entry points to the same neighboring AS. The value of the MULTI_EXIT_DISC attribute is a four octet unsigned number which is called a metric. All other factors being equal, the exit point with lower metric SHOULD be preferred. If received over EBGP, the MULTI_EXIT_DISC attribute MAY be propagated over IBGP to other BGP speakers within the same AS. The MULTI_EXIT_DISC attribute received from a neighboring AS MUST NOT be propagated to other neighboring ASs.

A BGP speaker MUST IMPLEMENT a mechanism based on local configuration which allows the MULTI_EXIT_DISC attribute to be removed from a route. This MAY be done prior to determining the degree of preference of the route and performing route selection (decision process phases 1 and 2).

An implementation MAY also (based on local configuration) alter the value of the MULTI_EXIT_DISC attribute received over EBGP. This MAY be done prior to determining the degree of preference of the route and performing route selection (decision process phases 1 and 2). See Section 9.1.2.2 for necessary restrictions on this.

5.1.5 LOCAL_PREF

LOCAL_PREF is a well-known attribute that SHALL be included in all UPDATE messages that a given BGP speaker sends to the other internal

peers. A BGP speaker SHALL calculate the degree of preference for each external route based on the locally configured policy, and include the degree of preference when advertising a route to its internal peers. The higher degree of preference MUST be preferred. A BGP speaker uses the degree of preference learned via LOCAL_PREF in its decision process (see Section 9.1.1).

A BGP speaker MUST NOT include this attribute in UPDATE messages that it sends to external peers, except for the case of BGP Confederations [RFC3065]. If it is contained in an UPDATE message that is received from an external peer, then this attribute MUST be ignored by the receiving speaker, except for the case of BGP Confederations [RF3065].

5.1.6 ATOMIC_AGGREGATE

ATOMIC_AGGREGATE is a well-known discretionary attribute.

When a BGP speaker aggregates several routes for the purpose of advertisement to a particular peer, the AS_PATH of the aggregated route normally includes an AS_SET formed from the set of ASs from which the aggregate was formed. In many cases the network administrator can determine that the aggregate can safely be advertised without the AS_SET and not form route loops.

If an aggregate excludes at least some of the AS numbers present in the AS_PATH of the routes that are aggregated as a result of dropping the AS_SET, the aggregated route, when advertised to the peer, SHOULD include the ATOMIC_AGGREGATE attribute.

A BGP speaker that receives a route with the ATOMIC_AGGREGATE attribute SHOULD NOT remove the attribute from the route when propagating it to other speakers.

A BGP speaker that receives a route with the ATOMIC_AGGREGATE attribute MUST NOT make any NLRI of that route more specific (as defined in 9.1.4) when advertising this route to other BGP speakers.

A BGP speaker that receives a route with the ATOMIC_AGGREGATE attribute needs to be cognizant of the fact that the actual path to destinations, as specified in the NLRI of the route, while having the loop-free property, may not be the path specified in the AS_PATH attribute of the route.

5.1.7 AGGREGATOR

AGGREGATOR is an optional transitive attribute which MAY be included in updates which are formed by aggregation (see Section 9.2.2.2). A BGP speaker which performs route aggregation MAY add the AGGREGATOR attribute which SHALL contain its own AS number and IP address. The IP address SHOULD be the same as the BGP Identifier of the speaker.

6. BGP Error Handling.

This section describes actions to be taken when errors are detected while processing BGP messages.

When any of the conditions described here are detected, a NOTIFICATION message with the indicated Error Code, Error Subcode, and Data fields is sent, and the BGP connection is closed, unless it is explicitly stated that no NOTIFICATION message is to be sent and the BGP connection is not to be closed. If no Error Subcode is specified, then a zero MUST be used.

The phrase "the BGP connection is closed" means that the TCP connection has been closed, the associated Adj-RIB-In has been cleared, and that all resources for that BGP connection have been deallocated. Entries in the Loc-RIB associated with the remote peer are marked as invalid. The fact that the routes have become invalid is passed to other BGP peers before the routes are deleted from the system.

Unless specified explicitly, the Data field of the NOTIFICATION message that is sent to indicate an error is empty.

6.1 Message Header error handling.

All errors detected while processing the Message Header are indicated by sending the NOTIFICATION message with Error Code Message Header Error. The Error Subcode elaborates on the specific nature of the error.

The expected value of the Marker field of the message header is all ones. If the Marker field of the message header is not as expected, then a synchronization error has occurred and the Error Subcode is set to Connection Not Synchronized.

If the Length field of the message header is less than 19 or greater

than 4096, or if the Length field of an OPEN message is less than the minimum length of the OPEN message, or if the Length field of an UPDATE message is less than the minimum length of the UPDATE message, or if the Length field of a KEEPALIVE message is not equal to 19, or if the Length field of a NOTIFICATION message is less than the minimum length of the NOTIFICATION message, then the Error Subcode is set to Bad Message Length. The Data field contains the erroneous Length field.

If the Type field of the message header is not recognized, then the Error Subcode is set to Bad Message Type. The Data field contains the erroneous Type field.

6.2 OPEN message error handling.

All errors detected while processing the OPEN message are indicated by sending the NOTIFICATION message with Error Code OPEN Message Error. The Error Subcode elaborates on the specific nature of the error.

If the version number contained in the Version field of the received OPEN message is not supported, then the Error Subcode is set to Unsupported Version Number. The Data field is a 2-octets unsigned integer, which indicates the largest locally supported version number less than the version the remote BGP peer bid (as indicated in the received OPEN message), or if the smallest locally supported version number is greater than the version the remote BGP peer bid, then the smallest locally supported version number.

If the Autonomous System field of the OPEN message is unacceptable, then the Error Subcode is set to Bad Peer AS. The determination of acceptable Autonomous System numbers is outside the scope of this protocol.

If the Hold Time field of the OPEN message is unacceptable, then the Error Subcode MUST be set to Unacceptable Hold Time. An implementation MUST reject Hold Time values of one or two seconds. An implementation MAY reject any proposed Hold Time. An implementation which accepts a Hold Time MUST use the negotiated value for the Hold Time.

If the BGP Identifier field of the OPEN message is syntactically incorrect, then the Error Subcode is set to Bad BGP Identifier. Syntactic correctness means that the BGP Identifier field represents a valid IP host address.

If one of the Optional Parameters in the OPEN message is not

recognized, then the Error Subcode is set to Unsupported Optional Parameters.

If one of the Optional Parameters in the OPEN message is recognized, but is malformed, then the Error Subcode is set to 0 (Unspecific).

6.3 UPDATE message error handling.

All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

Error checking of an UPDATE message begins by examining the path attributes. If the Withdrawn Routes Length or Total Attribute Length is too large (i.e., if Withdrawn Routes Length + Total Attribute Length + 23 exceeds the message Length), then the Error Subcode is set to Malformed Attribute List.

If any recognized attribute has Attribute Flags that conflict with the Attribute Type Code, then the Error Subcode is set to Attribute Flags Error. The Data field contains the erroneous attribute (type, length and value).

If any recognized attribute has Attribute Length that conflicts with the expected length (based on the attribute type code), then the Error Subcode is set to Attribute Length Error. The Data field contains the erroneous attribute (type, length and value).

If any of the mandatory well-known attributes are not present, then the Error Subcode is set to Missing Well-known Attribute. The Data field contains the Attribute Type Code of the missing well-known attribute.

If any of the mandatory well-known attributes are not recognized, then the Error Subcode is set to Unrecognized Well-known Attribute. The Data field contains the unrecognized attribute (type, length and value).

If the ORIGIN attribute has an undefined value, then the Error Subcode is set to Invalid Origin Attribute. The Data field contains the unrecognized attribute (type, length and value).

If the NEXT_HOP attribute field is syntactically incorrect, then the Error Subcode is set to Invalid NEXT_HOP Attribute. The Data field contains the incorrect attribute (type, length and value). Syntactic

correctness means that the NEXT_HOP attribute represents a valid IP host address.

The IP address in the NEXT_HOP MUST meet the following criteria to be considered semantically correct:

a) It MUST NOT be the IP address of the receiving speaker

b) In the case of an EBGP where the sender and receiver are one IP hop away from each other, either the IP address in the NEXT_HOP MUST be the sender's IP address (that is used to establish the BGP connection), or the interface associated with the NEXT_HOP IP address MUST share a common subnet with the receiving BGP speaker.

If the NEXT_HOP attribute is semantically incorrect, the error SHOULD be logged, and the route SHOULD be ignored. In this case, a NOTIFICATION message SHOULD NOT be sent, and connection SHOULD NOT be closed.

The AS_PATH attribute is checked for syntactic correctness. If the path is syntactically incorrect, then the Error Subcode is set to Malformed AS_PATH.

If the UPDATE message is received from an external peer, the local system MAY check whether the leftmost AS in the AS_PATH attribute is equal to the autonomous system number of the peer that sent the message. If the check determines that this is not the case, the Error Subcode is set to Malformed AS_PATH.

If an optional attribute is recognized, then the value of this attribute is checked. If an error is detected, the attribute is discarded, and the Error Subcode is set to Optional Attribute Error. The Data field contains the attribute (type, length and value).

If any attribute appears more than once in the UPDATE message, then the Error Subcode is set to Malformed Attribute List.

The NLRI field in the UPDATE message is checked for syntactic validity. If the field is syntactically incorrect, then the Error Subcode is set to Invalid Network Field.

If a prefix in the NLRI field is semantically incorrect (e.g., an unexpected multicast IP address), an error SHOULD be logged locally, and the prefix SHOULD be ignored.

An UPDATE message that contains correct path attributes, but no NLRI, SHALL be treated as a valid UPDATE message.

6.4 NOTIFICATION message error handling.

If a peer sends a NOTIFICATION message, and the receiver of the message detects an error in that message, the receiver can not use a NOTIFICATION message to report this error back to the peer. Any such error, such as an unrecognized Error Code or Error Subcode, SHOULD be noticed, logged locally, and brought to the attention of the administration of the peer. The means to do this, however, lies outside the scope of this document.

6.5 Hold Timer Expired error handling.

If a system does not receive successive KEEPALIVE and/or UPDATE and/or NOTIFICATION messages within the period specified in the Hold Time field of the OPEN message, then the NOTIFICATION message with Hold Timer Expired Error Code is sent and the BGP connection is closed.

6.6 Finite State Machine error handling.

Any error detected by the BGP Finite State Machine (e.g., receipt of an unexpected event) is indicated by sending the NOTIFICATION message with Error Code Finite State Machine Error.

6.7 Cease.

In absence of any fatal errors (that are indicated in this section), a BGP peer MAY choose at any given time to close its BGP connection by sending the NOTIFICATION message with Error Code Cease. However, the Cease NOTIFICATION message MUST NOT be used when a fatal error indicated by this section does exist.

A BGP speaker MAY support the ability to impose an (locally configured) upper bound on the number of address prefixes the speaker is willing to accept from a neighbor. When the upper bound is reached, the speaker (under control of local configuration) either (a) discards new address prefixes from the neighbor (while maintaining BGP connection with the neighbor), or (b) terminates the BGP connection with the neighbor. If the BGP speaker decides to terminate its BGP connection with a neighbor because the number of address prefixes received from the neighbor exceeds the locally configured upper

bound, then the speaker MUST send to the neighbor a NOTIFICATION message with the Error Code Cease.

6.8 BGP connection collision detection.

If a pair of BGP speakers try simultaneously to establish a BGP connection to each other, then two parallel connections between this pair of speakers might well be formed. If the source IP address used by one of these connections is the same as the destination IP address used by the other, and the destination IP address used by the first connection is the same as the source IP address used by the other, we refer to this situation as connection collision. Clearly in the presence of connection collision, one of these connections MUST be closed.

Based on the value of the BGP Identifier a convention is established for detecting which BGP connection is to be preserved when a collision does occur. The convention is to compare the BGP Identifiers of the peers involved in the collision and to retain only the connection initiated by the BGP speaker with the higher-valued BGP Identifier.

Upon receipt of an OPEN message, the local system MUST examine all of its connections that are in the OpenConfirm state. A BGP speaker MAY also examine connections in an OpenSent state if it knows the BGP Identifier of the peer by means outside of the protocol. If among these connections there is a connection to a remote BGP speaker whose BGP Identifier equals the one in the OPEN message, and this connection collides with the connection over which the OPEN message is received then the local system performs the following collision resolution procedure:

1. The BGP Identifier of the local system is compared to the BGP Identifier of the remote system (as specified in the OPEN message). Comparing BGP Identifiers is done by converting them to host byte order and treating them as (4-octet long) unsigned integers.

2. If the value of the local BGP Identifier is less than the remote one, the local system closes the BGP connection that already exists (the one that is already in the OpenConfirm state), and accepts the BGP connection initiated by the remote system.

3. Otherwise, the local system closes newly created BGP connection (the one associated with the newly received OPEN message), and continues to use the existing one (the one that is already in the OpenConfirm state).

Unless allowed via configuration, a connection collision with an existing BGP connection that is in Established state causes closing of the newly created connection.

Note that a connection collision can not be detected with connections that are in Idle, or Connect, or Active states.

Closing the BGP connection (that results from the collision resolution procedure) is accomplished by sending the NOTIFICATION message with the Error Code Cease.

7. BGP Version Negotiation

BGP speakers MAY negotiate the version of the protocol by making multiple attempts to open a BGP connection, starting with the highest version number each supports. If an open attempt fails with an Error Code OPEN Message Error, and an Error Subcode Unsupported Version Number, then the BGP speaker has available the version number it tried, the version number its peer tried, the version number passed by its peer in the NOTIFICATION message, and the version numbers that it supports. If the two peers do support one or more common versions, then this will allow them to rapidly determine the highest common version. In order to support BGP version negotiation, future versions of BGP MUST retain the format of the OPEN and NOTIFICATION messages.

8. BGP Finite State machine

This section specifies the BGP operation in terms of a Finite State Machine (FSM). The section falls into 2 parts:

1) Description of Events for the State machine (Section 8.1)
2) Description of the FSM (Section 8.2)

The data structures and FSM described in this document are conceptual and do not have to be implemented precisely as described here, as long as the implementations support the described functionality and their externally visible behavior is the same.

Session Attributes required for each connection are:

1) State
2) Connect Retry timer
3) Hold timer
4) Hold time

5) Keepalive timer
6) Keepalive time
7) Connect Retry Count
8) Connect Retry Initial Value

The optional Session attributes are listed below. These optional attributes may be supported either per connection or per local system:

1) Delay Open flag
2) Open Delay Timer
3) Perform automatic start flag
4) Perform automatic stop flag
5) Passive TCP establishment flag
6) Perform BGP peer oscillation damping flag
(which will be denoted as stop_peer_flap in text)
7) Idle Hold timer
8) Perform Collision detect in Established flag
9) Accept connections from un-configured peers
10) Track TCP state flag
11) Send NOTIFICATION without an OPEN flag

8.1 Events for the BGP FSM

8.1.1 Administrative Events

Please note that only Event 1 (manual start) and Event 2 (manual stop) are mandatory administrative events. All other administrative events are optional. The optional attributes do not have to be supported. However, if these attributes are supported, the state of the flags should be as indicated.

Event1: Manual start

Definition: Local system administrator manually starts peer connection.

Status: Mandatory

Optional attributes: Passive TCP establishment flag SHOULD not be set.

Event2: Manual stop

Definition: Local system administrator manually
stops the peer connection.

Status: Mandatory

Event3: Automatic start

Definition: Local system automatically starts the
BGP connection.

Status: Optional depending on local system.

Optional
attributes: 1) Perform automatic start flag SHOULD be set
if this event occurs.
2) if the passive Passive TCP establishment flag
is supported, it SHOULD not be set if this
event occurs.
3) if bgp peer oscillation damping is supported,
the BGP stop_peer_flap flag should not be set
when this event occurs.

Event4: Manual start with passive TCP flag

Definition: Local system administrator manually starts the peer
connection, but has the passive TCP establishment
enabled. The passive TCP establishment flag
indicates
that the peer will listen prior to
establishing the connection.

Status: Optional depending on local system.

Optional
attributes: 1) Passive TCP Establishment flag SHOULD be set.
if this event occurs.
2) If bgp peer oscilation damping is supported, the
stop_peer_flap flag should not be set when
this event occurs.

Event5: Automatic start with passive TCP flag

Definition: Local system automatically starts the
BGP connection with the passive flag
enabled. The passive flag indicates

that the peer will listen prior to
establishing a connection.

Status: Optional depending on local system use
of a passive connection and automatic start.

Optional
attributes: 1) Perform Automatic start flag SHOULD be set
2) Passive TCP establishment flag SHOULD be set
3) If the bgp peer oscillation flag is supported,
the stop_peer_flap flag SHOULD not be set.

Event6: Automatic start with bgp_stop_flap option set

Definition: Local system automatically starts the
BGP peer connection with peer oscillation
damping enabled. The exact method of damping
persistent peer oscillations is left up to the
implementation, and is outside the scope of
this document.

Status: Optional, used only if the bgp peer has enabled
bgp peer oscillation damping enabled with the
optional attribute settings below.

Optional
attributes: 1) Perform automatic start flag SHOULD be set
2) stop_peer_flap flag SHOULD be set
3) Passive TCP establishment flag SHOULD not be set
(cleared).

Event 7: Automatic start with bgp_stop_flap option set and passive
TCP establishment option set

Definition: Local system automatically starts the
BGP peer connection with peer oscillation
damping enabled and passive TCP establishment
enabled. The exact method of damping
persistent peer oscillations is left up to the
implementation, and is outside the scope of
this document.

Status: Optional, used only if the bgp peer has enabled
bgp peer oscillation damping with following optional
flags settings below.

Optional
attributes: 1) Perform automatic start flag SHOULD be set
2) stop_peer_flap flag SHOULD be set
3) Passive TCP establishment flag SHOULD be set

Event8: Automatic stop

Definition: Local system automatically stops the BGP connection.

An example of an automatic stop event is exceeding the number of prefixes for a given peer and the local system automatically disconnecting the peer.

Status: Optional depending on local system

Optional
attributes: 1) Peform automatic stop flag SHOULD Be set

8.1.2 Timer Events

Event9: Connect retry timer expires

Definition: An event generated when the Connect Retry timer expires.

Status: Mandatory

Event10: Hold timer expires

Definition: An event generated when the Hold Timer expires.

Status: Mandatory

Event11: Keepalive timer expires

Definition: An event generated when the Keepalive timer expires.
Status: Mandatory

Event12: Open Delay timer expires

Definition: An event generated when the Open Delay timer expires.

Status: Optional

Optional attributes: If this event occurs,
1) Delay Open flag SHOULD be set
2) Open Delay timer SHOULD be supported

Event13: Idle hold timer expires

Definition: An event generated when the Idle Hold Timer expires indicating that the session has completed waiting for a back-off period to prevent bgp peer oscillation.

The Idle Hold Timer is only used when the persistent peer oscillation damping function is enabled.

Implementations not implementing the presistent peer oscillation damping function may not have the Idle Hold Timer.

Status: Optional

Optional Attributes: If this event occurs:
1) stop_peer_flap flag SHOULD be set indicating support for persistent peer oscillation damping functions,
2) Idle Hold timer should be supported

8.1.3 TCP Connection based Events

Event14: TCP connection valid indication

Definition: Event indicating the local system reception of a TCP connection request with a valid source IP address and TCP port, and valid destination IP address and TCP Port. The definition of invalid source, and invalid destination IP address is left to the implementation.

BGP's destination port SHOULD be port
179 as defined by IANA.

TCP connection request is denoted by
the local system receiving a TCP SYN.

Status: Optional

Optional
Attributes: 1) The Track TCP state flag SHOULD be set if
this event occurs.

Event15: RCV TCP invalid indication

Definition: Event indicating the local system reception of
a TCP connection request with either
an invalid source address or port
number or an invalid destination
address or port number.

BGP destination port number SHOULD be 179
as defined by IANA.

Again, a TCP connection request
denoted by local system receiving a TCP
SYN.

Status: Optional

Optional
Attributes: 1) The Track TCP state should be set if this event
occurs.

Event16: TCP connection request Acknowledged

Definition: Event indicating the Local system's request
to establish a TCP connection to the remote
peer.

The local system's TCP session sent a TCP
SYN, and received a TCP SYN, ACK messages,
and Sent a TCP ACK.

Status: Mandatory

Event17: TCP connection confirmed

Definition: Event indicates that the local system receiving a confirmation that the TCP connection has been established by the remote site.

The remote peer's TCP engine sent a TCP SYN. The local peer's TCP engine sent a SYN, ACK message, and now has received a final ACK.

Status: Mandatory

Event18: TCP connection fails

Definition: Event indicates that the local system has received a TCP connection failure notice.

The remote BGP peer's TCP machine could have sent a FIN. The local peer would respond with a FIN-ACK. Another alternative is that the local peer indicated a timeout in the TCP session and downed the connection.

Status: Mandatory

8.1.4 BGP Messages based Events

Event19: BGPOpen

Definition: An event is generated when a valid OPEN message has been received.

Status: Mandatory

optional attributes: 1) Delay Open flag SHOULD not be set
2) Open Delay timer SHOULD not be running

Event20: BGPOpen with Open Delay Timer running

Definition: An event is generated when valid OPEN message has been received for a peer that has a successfully established transport connection and is currently delaying the sending of a BGP open message.

Status: Optional

Optional
attributes: 1) Delay Open Flag SHOULD be set
2) Open Delay Timer SHOULD be running.

Event21: BGPHeaderErr

Definition: An event is generated when a received BGP message header is not valid.

Status: Mandatory

Event22: BGPOpenMsgErr

Definition: An event is generated when an OPEN message has been received with errors.

Status: Mandatory

Event23: Open collision dump

Definition: An event generated administratively when a connection collision has been detected while processing an incoming OPEN message and this connection has been selected to disconnected. See Section 6.8 for more information on collision detection.

Event23 is an administrative based only implementation specific policy. This Event may occur if the FSM is implemented as two linked state machines.

Status: Optional, depending on local system

Optional
Attributes: If the state machine is to process this attribute in Established state,
1) Peform Collision detect in Established flag SHOULD be set.

Please note: The Open collision dump can occur

in Idle, Connect, Active, OpenSent, OpenConfirm without any optional flags being set.

Event24: NotifMsgVerErr

Definition: An event is generated when a NOTIFICATION message with "version error" is received.

Status: Mandatory

Event25: NotifMsg

Definition: An event is generated when a NOTIFICATION messages is received and the error code is anything but "version error".

Status: Mandatory

Event26: KeepAliveMsg

Definition: An event is generated when a KEEPALIVE message is received.

Status: Mandatory

Event27: UpdateMsg

Definition: An event is generated when a valid UPDATE message is received.

Status: Mandatory

Event28: UpdateMsgErr

Definition: An event is generated when an invalid UPDATE message is received.

Status: Mandatory

8.2 Description of FSM

8.2.1 FSM Definition

BGP MUST maintain a separate FSM for each configured peer, Each BGP peer paired in a potential connection unless configured to remain in the idle state, or configured to remain passive, will attempt to to connect to the other. For the purpose of this discussion, the active or connect side of the TCP connection (the side of a TCP connection sending the first TCP SYN packet) is called outgoing. The passive or listening side (the sender of the first SYN ACK) is called an incoming connection (see Section 8.2.1.1 on the terms active and passive below).

A BGP implementation MUST connect to and listen on TCP port 179 for incoming connections in addition to trying to connect to peers. For each incoming connection, a state machine MUST be instantiated. There exists a period in which the identity of the peer on the other end of an incoming connection is known but the BGP identifier is not known. During this time, both an incoming and an outgoing connection for the same configured peering may exist. This is referred to as a connection collision (see Section 6.8).

A BGP implementation will have at most one FSM for each configured peering plus one FSM for each incoming TCP connection for which the peer has not yet been identified. Each FSM corresponds to exactly one TCP connection.

There may be more than one connections between a pair of peers if the connections are configured to use a different pair of IP addresses. This is referred to as multiple "configured peerings" to the same peer.

8.2.1.1 Terms "active" and "passive"

The terms active and passive have been in our vocabulary for almost a decade and have proven useful. The words active and passive have slightly different meanings applied to a TCP connection or applied to a peer. There is only one active side and one passive side to any one TCP connection per the definition above and the state machine below. When a BGP speaker is configured active it may end up on either the active or passive side of the connection that eventually gets established. Once the TCP connection is completed, it doesn't matter which end was active and which end was passive and the only difference is which side of the TCP connection has port number 179.

8.2.1.2 FSM and collision detection

There is one FSM per BGP connection. Prior to determining what peer a connection is associated with there may be two connections for a given peer. There SHOULD be no more than one connection per peer. The collision detection identifies the case where there is more than one connection per peer and provides guidance for which connection to get rid of. When this occurs, the corresponding FSM for the connection that is closed SHOULD be disposed of.

8.2.1.3 FSM and Optional Attributes

Optional Attributes specify either flags that augment the normal processing of the BGP FSM, or optional timers. If a Optional attribute can be set on a system, the Events and the BGP FSM actions must be supported. For example, if the following options can be set in a BGP implementation: AutoStart and Passive TCP connection Establishment flag, then the events 3, 4 and 5 must be supported.

If an Optional attribute cannot be set (that is declared always off logically), the events supporting that set of options do not have to be supported.

8.2.1.4 FSM Event numbers

The Event numbers (1-28) utilized in this state machine description aid in specifying the behavior of the BGP state machine. Implementations MAY use these numbers to provide network management information. The exact form of the FSM and the FSM events is specific to each implementation.

8.2.2 Finite State Machine

Idle state:

Initially BGP is in the Idle state.

In this state BGP refuses all incoming BGP connections. No resources are allocated to the peer. In response to a manual start event(Event1) or an automatic start event(Event3), the local system:

- initializes all BGP resources,

- sets ConnectRetryCnt (the connect retry counter) to zero
- starts the Connect Retry timer with initial value,
- initiates a TCP connection to the other BGP peer,
- listens for a connection that may be initiated by the remote BGP peer, and
- changes its state to Connect.

The manual stop event (Event2) and Automatic stop event (Event 8) are ignored in the Idle state.

In response to a manual start event with the passive TCP connection flag (Event 4) or automatic start with the passive TCP connection flag (Event 5), the local system:

- initializes all BGP resources,
- sets ConnectRetryCnt (the connect retry counter) to zero,
- starts the Connect Retry timer with initial value,
- listens for a connection that may be initiated by the remote peer, and
- changes its state to Active.

The exact value of the ConnectRetry timer is a local matter, but it SHOULD be sufficiently large to allow TCP initialization.

If the persistent peer oscillation damping function is enabled, three additional events may occur within Idle state:

- Automatic start with peer_stop_flap set [Event6],
- Automatic start with peer_stop_flap set and passive TCP establishment flag set [Event7],
- Idle Hold Timer expired [Event 13].

The method of preventing persistent peer oscillation is outside the scope of this document.

Any other events [Events 9-12, 15-28] received in the Idle state does not cause change in the state of the local system.

Connect State:

In this state, BGP is waiting for the TCP connection to be completed.

The start events [Event 1, 3-7] are ignored in connect state.

In response to a manual stop event [Event2], the local system:
- drops the TCP connection,
- releases all BGP resources,
- sets ConnectRetryCnt (the connect retry count) to zero
- sets the Connect Retry timer to zero, and
- changes its state to Idle.

In response to the Connect Retry timer expires event [Event 9], the local system:
- drops the TCP connection,
- restarts the Connect Retry timer,
- stops the Open Delay timer and resets the timer to zero,
- initiates a TCP connection to the other BGP peer,
- continues to listen for a connection that may be initiated by the remote BGP peer, and
- stays in Connect state.

If the Open Delay timer expires [Event12] in the connect state, the local system:
- sends an OPEN message to its peer,
- sets the hold timer to a large value, and
- changes its state to OpenSent.

If the BGP port receives a valid TCP connection indication [Event 14], the TCP connection is processed and the connection remains in the Connect state.

If the TCP connection receives an invalid indication [Event 15]: the local system rejects the TCP connection and the connection remains in the Connect state.

If the TCP connection succeeds [Event 16 or Event 17], the local system checks the Delay Open flag prior to processing. If the Delay Open flag is set, the local system:
- sets the Connect Retry timer to zero,
- set the Open Delay timer to the initial value, and
- stays in the Connect state.

If the Delay Open flag is not set, the local system:
- sets the Connect Retry timer to zero,
- completes BGP initialization
- sends an OPEN message to its peer,
- sets hold timer to a large value, and
- changes its state to OpenSent.

A hold timer value of 4 minutes is suggested.

If the TCP connection fails [Event18], the local system checks

the Open Delay Timer. If the Open Delay timer is running, the local system:
- restarts the connect retry time with initial value,
- stops the Open Delay timer and resets value to zero,
- continues to listen for a connection that may be initiated by the remote BGP peer, and
- changes its state to Active.

If the open Delay timer is not running, the local system:
- sets the Connect Retry timer to zero,
- drops the TCP connection,
- releases all BGP resources, and
- changes its state to Idle.

If an OPEN message is received with the Open Delay timer is running [Event 20], the local system:
- sets the Connect Retry timer to zero,
- completes the BGP initialization,
- stops and clears the Open Delay timer (sets the value to zero),
- sends an OPEN message,
- sends a KEEPALIVE message,
- If the hold timer value is non-zero,
    - start the keepalive timer to inital value,
    - reset the hold timer to the negotiated value,

  else if hold timer value is zero,
    - reset the keepalive timer, and
    - reset the hold timer value to zero
- and changes its state to OpenConfirm.

If the value of the autonomous system field is the same as the local Autonomous System number, set the connection status to an internal connection; otherwise it is "external".

If BGP message header checking detects an error [Event 21] or OPEN message checking detects an error [Event 22] (see section 6.2), the local system:
- (optionally) If the Send Notification without Open flag is set, then the local system first sends a NOTIFICATION message with the appropriate error code, and then

- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping,
- and changes its state to Idle.

If a NOTIFICATION message is received with a version error[Event24], the local system checks the Open Delay timer.

If the Open Delay timer is running, the local system:
- sets the Connect Retry timer to zero,
- stops and reset the Open Delay timer (sets to zero),
- releases all BGP resources,
- drops the TCP connection, and
- changes its state to Idle.

If the Open Delay timer is not running, the local system:
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

In response to any other events [Events 8,10-11,13,19,23, 25-28] the local system:
- if the Connect Retry timer is running,
  stop and reset the Connect Retry timer (sets to zero),
- if the Open Delay timer is running,
  stop and reset the Open Delay timer (sets to zero),
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

Active State:

In this state BGP is trying to acquire a peer by listening for and accepting a TCP connection.

The start events [Event1, 3-7] are ignored in the Active state.

In response to a manual stop event[Event2], the local system:
- If the Open Delay timer is running and the Send NOTIFICATION without Open flag is set,
  the local system Sends a NOTIFICATION with a Cease,
- releases all BGP resources including
  - stopping the Open delay timer
- drops the TCP connection,
- sets ConnectRetryCnt (connect retry count) to zero
- sets the Connect Retry timer to zero, and
- changes its state to Idle.

In response the ConnectRetry timer expires event[Event9], the local system:

- restarts the Connect Retry timer (with initial value),
- initiates a TCP connection to the other BGP peer,
- Continues to listen for TCP connection that may be initiated by remote BGP peer, and
- changes its state to Connect.

If the local system has the Open Delay timer expired [Event12], the local system:

- sets the Connect Retry timer to zero,
- stops and clears the Open Delay timer (set to zero),
- completes the BGP initialization,
- sends the OPEN message to it's remote peer,
- sets its hold timer to a large value, and
- changes its state to OpenSent.

A hold timer value of 4 minutes is also suggested for this state transition.

If the local system receives a valid TCP indication [Event 14], the local system processes the TCP connection flags, and stays in Active state.

If the local system receives an invalid TCP indication [Event 15]: the local system rejects the TCP connection, and stays in the Active State.

In response to a TCP connection succeeds [Event 16 or Event 17], the local system checks the "Delay Open Flag" prior to processing. If the Delay Open flag is set, the local system

- o sets the Connect Retry timer to zero,
- o sets the Open Delay timer to the initial value, and
- o stays in the Active state.

-If the Delay Open flag is not set, the local system

- o sets the Connect Retry timer to zero,
- o completes the BGP initialization,
- o sends the OPEN message to it's peer,
- o sets its hold timer to a large value, and
- o changes its state to OpenSent.

A hold timer value of 4 minutes is suggested as a "large value" for the hold timer.

If the local system receives a TCP connection fails event [Event 18], the local system will:

- restart the Connect Retry timer (with initial value),
- stops and clears the Open Delay Timer (sets the value to zero),
- release all BGP resources
- Acknowledge the drop of TCP connection if
  TCP disconnect (send a FIN ACK),
- Increment ConnectRetryCnt (connect retry count) by 1, and
- optionally perform peer oscillation damping, and
- changes its state to Idle.

If an OPEN message is received with the Open Delay timer is running [Event 20], the local system

- sets the Connect Retry timer to zero,
- stops and clears the Open Delay timer
- completes the BGP initialization,
- sends an OPEN message,
- sends a KEEPALIVE message, and
- if the hold timer value is non-zero,
  - starts the keepalive timer to initial value,
  - resets the hold timer to the negotiated value,

  else if the hold timer is zero
  - resets the keepalive timer (set to zero),
  - resets the hold timer to zero,
- and changes its state to OpenConfirm.

If the value of the autonomous system field is the same as the local Autonomous System number, set the connection status to an internal connection; otherwise it is "external".

If BGP message header checking detects an error [Event 21] or OPEN message checking detects an error [Event 22] (see section 6.2), the local system:

- (optionally) sends NOTIFICATION message with the
  appropriate error code,
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping,
- and changes its state to Idle.

If a NOTIFICATION message is received with a version error[Event24], the local system checks the Open Delay timer. If the Open Delay timer is running, the local system:

- sets the Connect Retry timer to zero,
- stops and reset the Open Delay timer (sets to zero,

- releases all BGP resources,
- drops the TCP connection, and
- changes its state to Idle.

If the Open Delay timer is not running, the local system:
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

In response to any other event [Events 8,10-11,13,19,23,25-28], the local system:
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by one,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

OpenSent:

In this state BGP waits for an OPEN message from its peer.

The Start events [Event1, 3-7] are ignored in the OpenSent state.

If a manual stop event [Event 2] is issued in Open sent state, the local system:
- sends the NOTIFICATION with a cease,
- sets the Connect Retry timer to zero,
- release all BGP resources,
- drops the TCP connection,
- set ConnectRetryCnt (connect retry count) to zero, and
- changes its state to Idle.

If an automatic stop event [Event 8] is issued in OpenSent state, the local system:
- sends the NOTIFICATION with a cease,
- sets the Connect Retry timer to zero,
- release all the BGP resources
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If the Hold Timer expires[Event 10], the local system:

- send a NOTIFICATION message with error code Hold Timer Expired,
- set the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If a TCP indication is received for valid connection [Event 14] or TCP request aknowledgement [Event 16] is received, or a TCP connect confirm [Event 17] is received a second TCP session may be in progress. This second TCP session is tracked per the Connection Collision processing (Section 6.8) until an OPEN message is received.

A TCP connection for an invalid port [Event 15] is ignored.

If a TCP connection fails event [Event18] indication is received the local system:

- closes the BGP connection,
- restarts the Connect Retry timer,
- continues to listen for a connection that may be initiated by the remote BGP peer, and
- changes its state to Active.

When an OPEN message is received, all fields are checked for correctness. If there are no errors in the OPEN message [Event 19] the local system:

- resets the Open Delay timer to zero,
- sets the BGP Connect Retry timer to zero,
- sends a KEEPALIVE message and
- sets a KeepAlive timer (via the text below)
- sets the hold timer according to the negotiated value (see Section 4.2), and
- changes its state to OpenConfirm.

If the negotiated hold time value is zero, then the Hold and KeepAlive timers are not started. If the value of the Autonomous System field is the same as the local Autonomous System number, then the connection is an "internal" connection; otherwise, it is an "external" connection. (This will impact UPDATE processing as described below.)

If the BGP message header checking [Event21] or OPEN message check detects an error (see Section 6.2)[Event22], the local system:

- sends a NOTIFICATION message with appropriate error code,
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection
- increments the ConnectRetryCnt (connect retry cout) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

Collision detection mechanisms (Section 6.8) need to be applied when a valid BGP OPEN message is received [Event 19 or Event 20]. Please refer to Section 6.8 for the details of the comparison. An administrative collision detect is when BGP implementation determines by means outside the scope of this document that a connection collision has occurred.

If a connection in OpenSent is determined to be the connection that must be closed, an open collision dump [Event 23] is signaled to the state machine. If such an event is received in OpenSent, the local system:

- sends a NOTIFICATION with a Cease
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If a NOTIFICATION message is received with a version error[Event24], the local system:

- sets the Connect Retry timer to zero
- releases all BGP resources,
- drops the TCP connection,
- changes its state to Idle.

In response to any other event [Events 9, 11-13,20,25-28], the local system:

- sends the NOTIFICATION with the Error Code Finite state machine error,
- sets the Connect Retry timer to zero,
- releases all BGP resources
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

OpenConfirm State:

In this state BGP waits for a KEEPALIVE or NOTIFICATION message.

Any start event [Event1, 3-7] is ignored in the OpenConfirm state.

In response to a manual stop event[Event 2] initiated by the operator, the local system:
- sends the NOTIFICATION message with Cease,
- releases all BGP resources,
- drop the TCP connection,
- sets the ConnectRetryCnt (connect retry count) to zero
- sets the Connect Retry timer to zero, and
- changes its state to Idle.

In response to the Automatic stop event initiated by the system[Event 8], the local system:
- sends the NOTIFICATION message with Cease,
- sets the Connect Retry timer to zero,
- release all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If the Hold Timer expires before a KEEPALIVE message is received [Event 10], the local system:
- send the NOTIFICATION message with the error code set to Hold Time Expired,
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If the local system receives a KEEPALIVE timer expires event [Event 11], the system:
- sends a KEEPALIVE message,
- restarts the Keepalive timer, and
- remains in OpenConfirmed state.

In the event of TCP connection valid indication [Event 14], or TCP connection succeeding [Event 16 or Event 17] while in OpenConfirm, the local system needs to track the 2nd connection.

If a TCP connection is attempted to an invalid port [Event 15], the local system will ignore the second connection attempt.

If the local system receives a TCP connection fails event [Event 18] from the underlying TCP, or a NOTIFICATION message [Event 25] the local system:

- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If the local system receives a NOTIFICATION message [Event 24] with a version error, the local system:

- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection, and
- changes its state to Idle.

If the local system receives a valid OPEN message [Event 19], the collision detect function is processed per Section 6.8. If this connection is to be dropped due to connection collision, the local system:

- sends a NOTIFICATION with a Cease
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection (send TCP FIN),
- increments the ConnectRetryCnt by 1 (connect retry count),
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If an OPEN message is received, all fields are check for correctness. If the BGP message header checking [Event21] or OPEN message check detects an error (see Section 6.2)[Event22], the local system:

- sends a NOTIFICATION message with appropriate error code,

- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If during the processing of another OPEN message, the BGP implementation determines by means outside the scope of this document that a connection collision has occurred and this connection is to be closed, the local system will issue a open collision dump [Event 23]. When the local system receives a open collision dump event [Event 23], the local system:

- sends a NOTIFICATION with a Cease
- sets the Connect Retry timer to zero,
- releases all BGP resources
- drops all TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If the local system receives a KEEPALIVE message[Event 26],

- restarts the Hold timer, and
- changes its state to Established.

In response to any other event [Events 9, 12-13, 20, 27-28], the local system:

- sends a NOTIFICATION with a code of Finite State Machine Error,
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retrycount) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

Established State:

In the Established state BGP can exchange UPDATE, NOTFICATION, and KEEPALIVE messages with its peer.

Any start event (Event 1, 3-7) is ignored in the Established state.

In response to a manual stop event (initiated by an operator)[Event2], the local sytem:

- sends the NOTIFICATION message with Cease,
- sets the Connect Retry timer to zero,
- delete all routes associated with this connection,
- release BGP resources,
- drops TCP connection,
- sets ConnectRetryCnt (connect retry count) to zero (0), and
- changes its state to Idle.

In response to an automatic stop event initiated by the system (automatic) [Event8], the local system:

- sends a NOTIFICATION with Cease,
- sets the Connect Retry timer to zero
- deletes all routes associated with this connection,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

An example automatic stop event is exceeding the number of prefixes for a given peer and the local system automatically disconnecting the peer.

If the Hold timer expires [Event10], the local system:

- sends a NOTIFICATION message with Error Code Hold Timer Expired,
- sets the Connect Retry timer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If the KeepAlive timer expires [Event11], the local system sends a KEEPALIVE message, it restarts its KeepAlive timer, unless the negotiated Hold Time value is zero.

Each time the local system sends a KEEPALIVE or UPDATE message, it restarts its KeepAlive timer, unless the negotiated Hold Time value is zero.

A TCP connection indication [Event 14] received for a valid port will cause the 2nd connection to be tracked.

A TCP connection indications for invalid port [Event 15], will be ignored.

In response to a TCP connection succeeds [Event 16 or Event 17], the 2nd connection SHALL be tracked until it sends an OPEN message.

If a valid OPEN message [Event 19] is received, it will be checked to see if it collides (Section 6.8) with any other session. If the BGP implementation determines that this connection needs to be terminated, it will process an open collision dump event[Event 23]. If this session needs to be terminated, the connection will be terminated by:

- sends a NOTIFICATION with a Cease,
- sets the Connect Retry timer to zero,
- deletes all routes associated with this connection,
- releases all BGP resources,
- drops the TCP connection,
- increments ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

If the local system receives a NOTIFICATION message [Event24 or Event 25] or a TCP connections fails [Event18] from the underlying TCP, it:

- sets the Connect Retry timer to zero,
- deletes all routes associated with this connection,
- releases all the BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1, and
- changes its state to Idle.

If the local system receives a KEEPALIVE message [Event 26], the local system will:

- restarts its Hold Timer, if the negotiated Hold Time value is non-zero, and
- remain in the Established state.

If the local system receives an UPDATE message [Event27], the local system will:

- process the update packet
- restarts its Hold timer, if the negotiated Hold Time value is non-zero, and
- remain in the Established state.

If the local system receives an UPDATE message, and the UPDATE message error handling procedure (see Section 6.3) detects an error [Event28], the local system:

- sends a NOTIFICATION message with Update error,
- sets the Connect Retry timer to zero,
- delets all routes associated with this connection,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

In response to any other event [Events 9, 12-13, 20-22] the local system:

- sends a NOTIFICATION message with Error Code Finite State Machine Error,
- deletes all routes associated with this connection,
- sets the Connect Retry timer to zero
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCnt (connect retry count) by 1,
- optionally performs peer oscillation damping, and
- changes its state to Idle.

9. UPDATE Message Handling

An UPDATE message may be received only in the Established state. When an UPDATE message is received, each field is checked for validity as specified in Section 6.3.

If an optional non-transitive attribute is unrecognized, it is quietly ignored. If an optional transitive attribute is unrecognized, the Partial bit (the third high-order bit) in the attribute flags octet is set to 1, and the attribute is retained for propagation to other BGP speakers.

If an optional attribute is recognized, and has a valid value, then, depending on the type of the optional attribute, it is processed locally, retained, and updated, if necessary, for possible propagation to other BGP speakers.

If the UPDATE message contains a non-empty WITHDRAWN ROUTES field, the previously advertised routes whose destinations (expressed as IP prefixes) contained in this field SHALL be removed from the Adj-RIB-In. This BGP speaker SHALL run its Decision Process since the previously advertised route is no longer available for use.

If the UPDATE message contains a feasible route, the Adj-RIB-In will be updated with this route as follows: if the NLRI of the new route is identical to the one of the route currently stored in the Adj-RIB-In, then the new route SHALL replace the older route in the Adj-RIB-In, thus implicitly withdrawing the older route from service. Otherwise, if the Adj-RIB-In has no route with NLRI identical to the new route, the new route SHALL be placed in the Adj-RIB-In.

Once the BGP speaker updates the Adj-RIB-In, the speaker SHALL run its Decision Process.

9.1 Decision Process

The Decision Process selects routes for subsequent advertisement by applying the policies in the local Policy Information Base (PIB) to the routes stored in its Adj-RIBs-In. The output of the Decision Process is the set of routes that will be advertised to peers; the selected routes will be stored in the local speaker's Adj-RIB-Out according to policy.

The selection process is formalized by defining a function that takes the attribute of a given route as an argument and returns either (a) a non-negative integer denoting the degree of preference for the route, or (b) a value denoting that this route is ineligible to be installed in LocRib and will be excluded from the next phase of route selection.

The function that calculates the degree of preference for a given route SHALL NOT use as its inputs any of the following: the existence of other routes, the non-existence of other routes, or the path attributes of other routes. Route selection then consists of individual application of the degree of preference function to each feasible route, followed by the choice of the one with the highest degree of preference.

The Decision Process operates on routes contained in the Adj-RIB-In, and is responsible for:

- selection of routes to be used locally by the speaker

- selection of routes to be advertised to other BGP peers

- route aggregation and route information reduction

The Decision Process takes place in three distinct phases, each triggered by a different event:

a) Phase 1 is responsible for calculating the degree of preference for each route received from a peer.

b) Phase 2 is invoked on completion of phase 1. It is responsible for choosing the best route out of all those available for each distinct destination, and for installing each chosen route into the Loc-RIB.

c) Phase 3 is invoked after the Loc-RIB has been modified. It is responsible for disseminating routes in the Loc-RIB to each peer, according to the policies contained in the PIB. Route aggregation and information reduction can optionally be performed within this phase.

9.1.1 Phase 1: Calculation of Degree of Preference

The Phase 1 decision function is invoked whenever the local BGP speaker receives from a peer an UPDATE message that advertises a new route, a replacement route, or withdrawn routes.

The Phase 1 decision function is a separate process which completes when it has no further work to do.

The Phase 1 decision function locks an Adj-RIB-In prior to operating on any route contained within it, and unlocks it after operating on all new or unfeasible routes contained within it.

For each newly received or replacement feasible route, the local BGP speaker determines a degree of preference as follows:

If the route is learned from an internal peer, either the value of the LOCAL_PREF attribute is taken as the degree of preference, or the local system computes the degree of preference of the route based on preconfigured policy information. Note that the latter

(computing the degree of preference based on preconfigured policy information) may result in formation of persistent routing loops.

If the route is learned from an external peer, then the local BGP speaker computes the degree of preference based on preconfigured policy information. If the return value indicates that the route is ineligible, the route MAY NOT serve as an input to the next phase of route selection; otherwise the return value is used as the LOCAL_PREF value in any IBGP readvertisement.

The exact nature of this policy information and the computation involved is a local matter.

9.1.2 Phase 2: Route Selection

The Phase 2 decision function is invoked on completion of Phase 1. The Phase 2 function is a separate process which completes when it has no further work to do. The Phase 2 process considers all routes that are eligible in the Adj-RIBs-In.

The Phase 2 decision function is blocked from running while the Phase 3 decision function is in process. The Phase 2 function locks all Adj-RIBs-In prior to commencing its function, and unlocks them on completion.

If the NEXT_HOP attribute of a BGP route depicts an address that is not resolvable, or it would become unresolvable if the route was installed in the routing table the BGP route MUST be excluded from the Phase 2 decision function.

If the AS_PATH attribute of a BGP route contains an AS loop, the BGP route should be excluded from the Phase 2 decision function. AS loop detection is done by scanning the full AS path (as specified in the AS_PATH attribute), and checking that the autonomous system number of the local system does not appear in the AS path. Operations of a BGP speaker that is configured to accept routes with its own autonomous system number in the AS path are outside the scope of this document.

It is critical that BGP speakers within an AS do not make conflicting decisions regarding route selection that would cause forwarding loops to occur.

For each set of destinations for which a feasible route exists in the Adj-RIBs-In, the local BGP speaker identifies the route that has:

a) the highest degree of preference of any route to the same set

of destinations, or

b) is the only route to that destination, or

c) is selected as a result of the Phase 2 tie breaking rules specified in 9.1.2.2.

The local speaker SHALL then install that route in the Loc-RIB, replacing any route to the same destination that is currently being held in the Loc-RIB. When the new BGP route is installed in the Routing Table, care must be taken to ensure that existing routes to the same destination that are now considered invalid are removed from the Routing Table. Whether or not the new BGP route replaces an existing non-BGP route in the Routing Table depends on the policy configured on the BGP speaker.

The local speaker MUST determine the immediate next-hop address from the NEXT_HOP attribute of the selected route (see Section 5.1.3). If either the immediate next hop or the IGP cost to the NEXT_HOP (where the NEXT_HOP is resolved through an IGP route) changes, Phase 2 Route Selection MUST be performed again.

Notice that even though BGP routes do not have to be installed in the Routing Table with the immediate next hop(s), implementations MUST take care that before any packets are forwarded along a BGP route, its associated NEXT_HOP address is resolved to the immediate (directly connected) next-hop address and this address (or multiple addresses) is finally used for actual packet forwarding.

Unresolvable routes SHALL be removed from the Loc-RIB and the routing table. However, corresponding unresolvable routes SHOULD be kept in the Adj-RIBs-In (in case they become resolvable).

9.1.2.1 Route Resolvability Condition

As indicated in Section 9.1.2, BGP speakers SHOULD exclude unresolvable routes from the Phase 2 decision. This ensures that only valid routes are installed in Loc-RIB and the Routing Table.

The route resolvability condition is defined as follows.

1. A route Rte1, referencing only the intermediate network address, is considered resolvable if the Routing Table contains at least one resolvable route Rte2 that matches Rte1's intermediate network address and is not recursively resolved (directly or indirectly) through Rte1. If multiple matching routes are available,

only the longest matching route SHOULD be considered.

2. Routes referencing interfaces (with or without intermediate addresses) are considered resolvable if the state of the referenced interface is up and IP processing is enabled on this interface.

BGP routes do not refer to interfaces, but can be resolved through the routes in the Routing Table that can be of both types (those that specify interfaces or those that do not). IGP routes and routes to directly connected networks are expected to specify the outbound interface. Static routes can specify the outbound interface, or the intermediate address, or both.

Note that a BGP route is considered unresolvable not only in situations where the BGP speaker's Routing Table contains no route matching the BGP route's NEXT_HOP. Mutually recursive routes (routes resolving each other or themselves), also fail the resolvability check.

It is also important that implementations do not consider feasible routes that would become unresolvable if they were installed in the Routing Table even if their NEXT_HOPs are resolvable using the current contents of the Routing Table (an example of such routes would be mutually recursive routes). This check ensures that a BGP speaker does not install in the Routing Table routes that will be removed and not used by the speaker. Therefore, in addition to local Routing Table stability, this check also improves behavior of the protocol in the network.

Whenever a BGP speaker identifies a route that fails the resolvability check because of mutual recursion, an error message SHOULD be logged.

9.1.2.2 Breaking Ties (Phase 2)

In its Adj-RIBs-In a BGP speaker may have several routes to the same destination that have the same degree of preference. The local speaker can select only one of these routes for inclusion in the associated Loc-RIB. The local speaker considers all routes with the same degrees of preference, both those received from internal peers, and those received from external peers.

The following tie-breaking procedure assumes that for each candidate route all the BGP speakers within an autonomous system can ascertain the cost of a path (interior distance) to the address depicted by the

NEXT_HOP attribute of the route, and follow the same route selection algorithm.

The tie-breaking algorithm begins by considering all equally preferable routes to the same destination, and then selects routes to be removed from consideration. The algorithm terminates as soon as only one route remains in consideration. The criteria MUST be applied in the order specified.

Several of the criteria are described using pseudo-code. Note that the pseudo-code shown was chosen for clarity, not efficiency. It is not intended to specify any particular implementation. BGP implementations MAY use any algorithm which produces the same results as those described here.

a) Remove from consideration all routes which are not tied for having the smallest number of AS numbers present in their AS_PATH attributes. Note, that when counting this number, an AS_SET counts as 1, no matter how many ASs are in the set.

b) Remove from consideration all routes which are not tied for having the lowest Origin number in their Origin attribute.

c) Remove from consideration routes with less-preferred MULTI_EXIT_DISC attributes. MULTI_EXIT_DISC is only comparable between routes learned from the same neighboring AS (the neighboring AS is determined from the AS_PATH attribute). Routes which do not have the MULTI_EXIT_DISC attribute are considered to have the lowest possible MULTI_EXIT_DISC value.

This is also described in the following procedure:

```
for m = all routes still under consideration
    for n = all routes still under consideration
        if (neighborAS(m) == neighborAS(n)) and (MED(n) < MED(m))
            remove route m from consideration
```

In the pseudo-code above, MED(n) is a function which returns the value of route n's MULTI_EXIT_DISC attribute. If route n has no MULTI_EXIT_DISC attribute, the function returns the lowest possible MULTI_EXIT_DISC value, i.e. 0.

Similarly, neighborAS(n) is a function which returns the neighbor AS from which the route was received. If the route is learned via IBGP, and the other IBGP speaker didn't originate the route, it is the neighbor AS from which the other IBGP speaker learned the route. If the route is learned via IBGP, and the other IBGP speaker originated the route, it is the local AS.

If a MULTI_EXIT_DISC attribute is removed before re-advertising a route into IBGP, then comparison based on the received EBGP MULTI_EXIT_DISC attribute MAY still be performed. If an implementation chooses to remove MULTI_EXIT_DISC, then the optional comparison on MULTI_EXIT_DISC if performed at all MUST be performed only among EBGP learned routes. The best EBGP learned route may then be compared with IBGP learned routes after the removal of the MULTI_EXIT_DISC attribute. If MULTI_EXIT_DISC is removed from a subset of EBGP learned routes and the selected "best" EBGP learned route will not have MULTI_EXIT_DISC removed, then the MULTI_EXIT_DISC must be used in the comparison with IBGP learned routes. For IBGP learned routes the MULTI_EXIT_DISC MUST be used in route comparisons which reach this step in the decision process. Including the MULTI_EXIT_DISC of an EBGP learned route in the comparison with an IBGP learned route, then removing the MULTI_EXIT_DISC atribute and advertising the route has been proven to cause route loops.

d) If at least one of the candidate routes was received via EBGP, remove from consideration all routes which were received via IBGP.

e) Remove from consideration any routes with less-preferred interior cost. The interior cost of a route is determined by calculating the metric to the NEXT_HOP for the route using the Routing Table. If the NEXT_HOP hop for a route is reachable, but no cost can be determined, then this step should be skipped (equivalently, consider all routes to have equal costs).

This is also described in the following procedure.

```
      for m = all routes still under consideration
          for n = all routes in still under consideration
              if (cost(n) is lower than cost(m))
                  remove m from consideration
```

In the pseudo-code above, cost(n) is a function which returns the cost of the path (interior distance) to the address given in the NEXT_HOP attribute of the route.

f) Remove from consideration all routes other than the route that was advertised by the BGP speaker whose BGP Identifier has the lowest value.

g) Prefer the route received from the lowest peer address.

9.1.3 Phase 3: Route Dissemination

The Phase 3 decision function is invoked on completion of Phase 2, or when any of the following events occur:

a) when routes in the Loc-RIB to local destinations have changed

b) when locally generated routes learned by means outside of BGP have changed

c) when a new BGP speaker - BGP speaker connection has been established

The Phase 3 function is a separate process which completes when it has no further work to do. The Phase 3 Routing Decision function is blocked from running while the Phase 2 decision function is in process.

All routes in the Loc-RIB are processed into Adj-RIBs-Out according to configured policy. This policy MAY exclude a route in the Loc-RIB from being installed in a particular Adj-RIB-Out. A route SHALL NOT be installed in the Adj-Rib-Out unless the destination and NEXT_HOP described by this route may be forwarded appropriately by the Routing Table. If a route in Loc-RIB is excluded from a particular Adj-RIB-Out the previously advertised route in that Adj-RIB-Out MUST be withdrawn from service by means of an UPDATE message (see 9.2).

Route aggregation and information reduction techniques (see 9.2.2.1) may optionally be applied.

Any local policy which results in routes being added to an Adj-RIB-Out without also being added to the local BGP speaker's forwarding table, is outside the scope of this document.

When the updating of the Adj-RIBs-Out and the Routing Table is complete, the local BGP speaker runs the Update-Send process of 9.2.

9.1.4 Overlapping Routes

A BGP speaker may transmit routes with overlapping Network Layer Reachability Information (NLRI) to another BGP speaker. NLRI overlap occurs when a set of destinations are identified in non-matching multiple routes. Since BGP encodes NLRI using IP prefixes, overlap will always exhibit subset relationships. A route describing a smaller set of destinations (a longer prefix) is said to be more specific

than a route describing a larger set of destinations (a shorter prefix); similarly, a route describing a larger set of destinations is said to be less specific than a route describing a smaller set of destinations.

The precedence relationship effectively decomposes less specific routes into two parts:

- a set of destinations described only by the less specific route, and

- a set of destinations described by the overlap of the less specific and the more specific routes

When overlapping routes are present in the same Adj-RIB-In, the more specific route takes precedence, in order from more specific to least specific.

The set of destinations described by the overlap represents a portion of the less specific route that is feasible, but is not currently in use. If a more specific route is later withdrawn, the set of destinations described by the overlap will still be reachable using the less specific route.

If a BGP speaker receives overlapping routes, the Decision Process MUST consider both routes based on the configured acceptance policy. If both a less and a more specific route are accepted, then the Decision Process MUST either install both the less and the more specific routes or it MUST aggregate the two routes and install the aggregated route, provided that both routes have the same value of the NEXT_HOP attribute.

If a BGP speaker chooses to aggregate, then it SHOULD either include all AS used to form the aggreagate in an AS_SET or add the ATOMIC_AGGREGATE attribute to the route. This attribute is now primarily informational. With the elimination of IP routing protocols that do not support classless routing and the elimination of router and host implementations that do not support classless routing, there is no longer a need to deaggregate. Routes SHOULD NOT be de-aggregated. A route that carries ATOMIC_AGGREGATE attribute in particular MUST NOT be de-aggregated. That is, the NLRI of this route can not be made more specific. Forwarding along such a route does not guarantee that IP packets will actually traverse only ASs listed in the AS_PATH attribute of the route.

9.2 Update-Send Process

The Update-Send process is responsible for advertising UPDATE messages to all peers. For example, it distributes the routes chosen by the Decision Process to other BGP speakers which may be located in either the same autonomous system or a neighboring autonomous system.

When a BGP speaker receives an UPDATE message from an internal peer, the receiving BGP speaker SHALL NOT re-distribute the routing information contained in that UPDATE message to other internal peers, unless the speaker acts as a BGP Route Reflector [RFC2796].

As part of Phase 3 of the route selection process, the BGP speaker has updated its Adj-RIBs-Out. All newly installed routes and all newly unfeasible routes for which there is no replacement route SHALL be advertised to its peers by means of an UPDATE message.

A BGP speaker SHOULD NOT advertise a given feasible BGP route from its Adj-RIB-Out if it would produce an UPDATE message containing the same BGP route as was previously advertised.

Any routes in the Loc-RIB marked as unfeasible SHALL be removed. Changes to the reachable destinations within its own autonomous system SHALL also be advertised in an UPDATE message.

If due to the limits on the maximum size of an UPDATE message (see Section 4) a single route doesn't fit into the message, the BGP speaker MUST not advertise the route to its peers and MAY choose to log an error locally.

9.2.1 Controlling Routing Traffic Overhead

The BGP protocol constrains the amount of routing traffic (that is, UPDATE messages) in order to limit both the link bandwidth needed to advertise UPDATE messages and the processing power needed by the Decision Process to digest the information contained in the UPDATE messages.

9.2.1.1 Frequency of Route Advertisement

The parameter MinRouteAdvertisementInterval determines the minimum

amount of time that must elapse between advertisement and/or withdrawal of routes to a particular destination by a BGP speaker to a peer. This rate limiting procedure applies on a per-destination basis, although the value of MinRouteAdvertisementInterval is set on a per BGP peer basis.

Two UPDATE messages sent by a BGP speaker to a peer that advertise feasible routes and/or withdrawal of unfeasible routes to some common set of destinations MUST be separated by at least MinRouteAdvertisementInterval. Clearly, this can only be achieved precisely by keeping a separate timer for each common set of destinations. This would be unwarranted overhead. Any technique which ensures that the interval between two UPDATE messages sent from a BGP speaker to a peer that advertise feasible routes and/or withdrawal of unfeasible routes to some common set of destinations will be at least MinRouteAdvertisementInterval, and will also ensure a constant upper bound on the interval is acceptable.

Since fast convergence is needed within an autonomous system, either (a) the MinRouteAdvertisementInterval used for internal peers SHOULD be shorter than the MinRouteAdvertisementInterval used for external peers, or (b) the procedure describe in this section SHOULD NOT apply for routes sent to internal peers.

This procedure does not limit the rate of route selection, but only the rate of route advertisement. If new routes are selected multiple times while awaiting the expiration of MinRouteAdvertisementInterval, the last route selected SHALL be advertised at the end of MinRouteAdvertisementInterval.

9.2.1.2 Frequency of Route Origination

The parameter MinASOriginationInterval determines the minimum amount of time that must elapse between successive advertisements of UPDATE messages that report changes within the advertising BGP speaker's own autonomous systems.

9.2.2 Efficient Organization of Routing Information

Having selected the routing information which it will advertise, a BGP speaker may avail itself of several methods to organize this information in an efficient manner.

9.2.2.1 Information Reduction

Information reduction may imply a reduction in granularity of policy control - after information is collapsed, the same policies will apply to all destinations and paths in the equivalence class.

The Decision Process may optionally reduce the amount of information that it will place in the Adj-RIBs-Out by any of the following methods:

a) Network Layer Reachability Information (NLRI):

Destination IP addresses can be represented as IP address prefixes. In cases where there is a correspondence between the address structure and the systems under control of an autonomous system administrator, it will be possible to reduce the size of the NLRI carried in the UPDATE messages.

b) AS_PATHs:

AS path information can be represented as ordered AS_SEQUENCEs or unordered AS_SETs. AS_SETs are used in the route aggregation algorithm described in 9.2.2.2. They reduce the size of the AS_PATH information by listing each AS number only once, regardless of how many times it may have appeared in multiple AS_PATHs that were aggregated.

An AS_SET implies that the destinations listed in the NLRI can be reached through paths that traverse at least some of the constituent autonomous systems. AS_SETs provide sufficient information to avoid routing information looping; however their use may prune potentially feasible paths, since such paths are no longer listed individually as in the form of AS_SEQUENCEs. In practice this is not likely to be a problem, since once an IP packet arrives at the edge of a group of autonomous systems, the BGP speaker at that point is likely to have more detailed path information and can distinguish individual paths to destinations.

9.2.2.2 Aggregating Routing Information

Aggregation is the process of combining the characteristics of several different routes in such a way that a single route can be advertised. Aggregation can occur as part of the decision process to reduce the amount of routing information that will be placed in the Adj-RIBs-Out.

Aggregation reduces the amount of information that a BGP speaker must store and exchange with other BGP speakers. Routes can be aggregated by applying the following procedure separately to path attributes of like type and to the Network Layer Reachability Information.

Routes that have different MULTI_EXIT_DISC attribute SHALL NOT be aggregated.

Path attributes that have different type codes can not be aggregated together. Path attributes of the same type code may be aggregated, according to the following rules:

NEXT_HOP:
When aggregating routes that have different NEXT_HOP attribute, the NEXT_HOP attribute of the aggregated route SHALL identify an interface on the BGP speaker that performs the aggregation.

ORIGIN attribute:
If at least one route among routes that are aggregated has ORIGIN with the value INCOMPLETE, then the aggregated route MUST have the ORIGIN attribute with the value INCOMPLETE. Otherwise, if at least one route among routes that are aggregated has ORIGIN with the value EGP, then the aggregated route MUST have the origin attribute with the value EGP. In all other case the value of the ORIGIN attribute of the aggregated route is IGP.

AS_PATH attribute:
If routes to be aggregated have identical AS_PATH attributes, then the aggregated route has the same AS_PATH attribute as each individual route.

For the purpose of aggregating AS_PATH attributes we model each AS within the AS_PATH attribute as a tuple <type, value>, where "type" identifies a type of the path segment the AS belongs to (e.g. AS_SEQUENCE, AS_SET), and "value" is the AS number. If the routes to be aggregated have different AS_PATH attributes, then the aggregated AS_PATH attribute SHALL satisfy all of the following conditions:

- all tuples of type AS_SEQUENCE in the aggregated AS_PATH SHALL appear in all of the AS_PATH in the initial set of routes to be aggregated.

- all tuples of type AS_SET in the aggregated AS_PATH SHALL appear in at least one of the AS_PATH in the initial set (they may appear as either AS_SET or AS_SEQUENCE types).

- for any tuple X of type AS_SEQUENCE in the aggregated AS_PATH which precedes tuple Y in the aggregated AS_PATH, X precedes Y in each AS_PATH in the initial set which contains Y, regardless of the type of Y.

- No tuple of type AS_SET with the same value SHALL appear more than once in the aggregated AS_PATH.

- Multiple tuples of type AS_SEQUENCE with the same value may appear in the aggregated AS_PATH only when adjacent to another tuple of the same type and value.

An implementation may choose any algorithm which conforms to these rules. At a minimum a conformant implementation SHALL be able to perform the following algorithm that meets all of the above conditions:

- determine the longest leading sequence of tuples (as defined above) common to all the AS_PATH attributes of the routes to be aggregated. Make this sequence the leading sequence of the aggregated AS_PATH attribute.

- set the type of the rest of the tuples from the AS_PATH attributes of the routes to be aggregated to AS_SET, and append them to the aggregated AS_PATH attribute.

- if the aggregated AS_PATH has more than one tuple with the same value (regardless of tuple's type), eliminate all, but one such tuple by deleting tuples of the type AS_SET from the aggregated AS_PATH attribute.

- for each pair of adjacent tuples in the aggregated AS_PATH, if both tuples have the same type, merge them together, as long as doing so will not cause a segment with length greater than 255 to be generated.

Appendix F, Section F.6 presents another algorithm that satisfies the conditions and allows for more complex policy configurations.

ATOMIC_AGGREGATE:

If at least one of the routes to be aggregated has ATOMIC_AGGREGATE path attribute, then the aggregated route SHALL have this attribute as well.

AGGREGATOR:

Any AGGREGATOR attributes from the routes to be aggregated MUST NOT be included in the aggregated route. The BGP speaker

performing the route aggregation MAY attach a new AGGREGATOR attribute (see Section 5.1.7).

9.3 Route Selection Criteria

Generally speaking, additional rules for comparing routes among several alternatives are outside the scope of this document. There are two exceptions:

- If the local AS appears in the AS path of the new route being considered, then that new route can not be viewed as better than any other route (provided that the speaker is configured to accept such routes). If such a route were ever used, a routing loop could result.

- In order to achieve successful distributed operation, only routes with a likelihood of stability can be chosen. Thus, an AS SHOULD avoid using unstable routes, and it SHOULD NOT make rapid spontaneous changes to its choice of route. Quantifying the terms "unstable" and "rapid" in the previous sentence will require experience, but the principle is clear.

Care must be taken to ensure that BGP speakers in the same AS do not make inconsistent decisions.

9.4 Originating BGP routes

A BGP speaker may originate BGP routes by injecting routing information acquired by some other means (e.g. via an IGP) into BGP. A BGP speaker that originates BGP routes assigns the degree of preference to these routes by passing them through the Decision Process (see Section 9.1). These routes MAY also be distributed to other BGP speakers within the local AS as part of the update process (see Section 9.2). The decision whether to distribute non-BGP acquired routes within an AS via BGP or not depends on the environment within the AS (e.g. type of IGP) and SHOULD be controlled via configuration.

10 BGP Timers

BGP employs five timers: ConnectRetry (see Section 8), Hold Time (see Section 4.2), KeepAlive (see Section 8), MinASOriginationInterval (see Section 9.2.1.2), and MinRouteAdvertisementInterval (see Section 9.2.1.1).

The suggested default value for the ConnectRetry timer is 120 seconds.

The suggested default value for the Hold Time is 90 seconds.

The suggested default value for the KeepAlive timer is 1/3 of the Hold Time.

The suggested default value for the MinASOriginationInterval is 15 seconds.

The suggested default value for the MinRouteAdvertisementInterval is 30 seconds.

An implementation of BGP MUST allow the Hold Time timer to be configurable on a per peer basis, and MAY allow the other timers to be configurable.

To minimize the likelihood that the distribution of BGP messages by a given BGP speaker will contain peaks, jitter SHOULD be applied to the timers associated with MinASOriginationInterval, KeepAlive, MinRouteAdvertisementInterval, and ConnectRetry. A given BGP speaker MAY apply the same jitter to each of these quantities regardless of the destinations to which the updates are being sent; that is, jitter need not be configured on a "per peer" basis.

The suggested default amount of jitter SHALL be determined by multiplying the base value of the appropriate timer by a random factor which is uniformly distributed in the range from 0.75 to 1.0. A new random value SHOULD be picked each time the timer is set. The range of the jitter random value MAY be configurable.

Appendix A. Comparison with RFC1771

There are numerous editorial changes (too many to list here).

The following list the technical changes:

- Changes to reflect the usages of such features as TCP MD5 [RFC2385], BGP Route Reflectors [RFC2796], BGP Confederations [RFC3065], and BGP Route Refresh [RFC2918].

- Clarification on the use of the BGP Identifier in the AGGREGATOR attribute.

- Procedures for imposing an upper bound on the number of prefixes

that a BGP speaker would accept from a peer.

The ability of a BGP speaker to include more than one instance of its own AS in the AS_PATH attribute for the purpose of inter-AS traffic engineering.

Clarifications on the various types of NEXT_HOPs.

Clarifications to the use of the ATOMIC_AGGREGATE attribute.

The relationship between the immediate next hop, and the next hop as specified in the NEXT_HOP path attribute.

Clarifications on the tie-breaking procedures.

Clarifications on the frequency of route advertisements.

Optional Parameter Type 1 (Authentication Information) has been deprecated.

UPDATE Message Error subcode 7 (AS Routing Loop) has been deprecated.

OPEN Message Error subcode 5 (Authentication Failure) has been deprecated.

Use of the Marker field for authentication has been deprecated.

Implementations MUST support TCP MD5 [RFC2385] for authentication.

Appendix B. Comparison with RFC1267

All the changes listed in Appendix A, plus the following.

BGP-4 is capable of operating in an environment where a set of reachable destinations may be expressed via a single IP prefix. The concept of network classes, or subnetting is foreign to BGP-4. To accommodate these capabilities BGP-4 changes semantics and encoding associated with the AS_PATH attribute. New text has been added to define semantics associated with IP prefixes. These abilities allow BGP-4 to support the proposed supernetting scheme [9].

To simplify configuration this version introduces a new attribute, LOCAL_PREF, that facilitates route selection procedures.

The INTER_AS_METRIC attribute has been renamed to be MULTI_EXIT_DISC.

A new attribute, ATOMIC_AGGREGATE, has been introduced to insure that certain aggregates are not de-aggregated. Another new attribute, AGGREGATOR, can be added to aggregate routes in order to advertise which AS and which BGP speaker within that AS caused the aggregation.

To insure that Hold Timers are symmetric, the Hold Time is now negotiated on a per-connection basis. Hold Times of zero are now supported.

Appendix C. Comparison with RFC 1163

All of the changes listed in Appendices A and B, plus the following.

To detect and recover from BGP connection collision, a new field (BGP Identifier) has been added to the OPEN message. New text (Section 6.8) has been added to specify the procedure for detecting and recovering from collision.

The new document no longer restricts the router that is passed in the NEXT_HOP path attribute to be part of the same Autonomous System as the BGP Speaker.

New document optimizes and simplifies the exchange of the information about previously reachable routes.

Appendix D. Comparison with RFC 1105

All of the changes listed in Appendices A, B and C, plus the following.

Minor changes to the RFC1105 Finite State Machine were necessary to accommodate the TCP user interface provided by 4.3 BSD.

The notion of Up/Down/Horizontal relations present in RFC1105 has been removed from the protocol.

The changes in the message format from RFC1105 are as follows:

1. The Hold Time field has been removed from the BGP header and added to the OPEN message.

2. The version field has been removed from the BGP header and added to the OPEN message.

3. The Link Type field has been removed from the OPEN message.

4. The OPEN CONFIRM message has been eliminated and replaced with implicit confirmation provided by the KEEPALIVE message.

5. The format of the UPDATE message has been changed significantly. New fields were added to the UPDATE message to support multiple path attributes.

6. The Marker field has been expanded and its role broadened to support authentication.

Note that quite often BGP, as specified in RFC 1105, is referred to as BGP-1, BGP, as specified in RFC 1163, is referred to as BGP-2, BGP, as specified in RFC1267 is referred to as BGP-3, and BGP, as specified in this document is referred to as BGP-4.

Appendix E. TCP options that may be used with BGP

If a local system TCP user interface supports TCP PUSH function, then each BGP message SHOULD be transmitted with PUSH flag set. Setting PUSH flag forces BGP messages to be transmitted promptly to the receiver.

If a local system TCP user interface supports setting of the DSCP field [RFC2474] for TCP connections, then the TCP connection used by BGP SHOULD be opened with bits 0-2 of the DSCP field set to 110 (binary).

Appendix F. Implementation Recommendations

This section presents some implementation recommendations.

Appendix F.1 Multiple Networks Per Message

The BGP protocol allows for multiple address prefixes with the same path attributes to be specified in one message. Making use of this capability is highly recommended. With one address prefix per message there is a substantial increase in overhead in the receiver. Not only does the system overhead increase due to the reception of multiple messages, but the overhead of scanning the routing table for updates to BGP peers and other routing protocols (and sending the associated messages) is incurred multiple times as well.

One method of building messages containing many address prefixes per a path attribute set from a routing table that is not organized on a per path attribute set basis is to build many messages as the routing table is scanned. As each address prefix is processed, a message for the associated set of path attributes is allocated, if it does not exist, and the new address prefix is added to it. If such a message exists, the new address prefix is just appended to it. If the message lacks the space to hold the new address prefix, it is transmitted, a new message is allocated, and the new address prefix is inserted into the new message. When the entire routing table has been scanned, all allocated messages are sent and their resources released. Maximum compression is achieved when all the destinations covered by the address prefixes share a common set of path attributes making it possible to send many address prefixes in one 4096-byte message.

When peering with a BGP implementation that does not compress multiple address prefixes into one message, it may be necessary to take steps to reduce the overhead from the flood of data received when a peer is acquired or a significant network topology change occurs. One method of doing this is to limit the rate of updates. This will eliminate the redundant scanning of the routing table to provide flash updates for BGP peers and other routing protocols. A disadvantage of this approach is that it increases the propagation latency of routing information. By choosing a minimum flash update interval that is not much greater than the time it takes to process the multiple messages this latency should be minimized. A better method would be to read all received messages before sending updates.

Appendix F.2 Reducing route flapping

To avoid excessive route flapping a BGP speaker which needs to withdraw a destination and send an update about a more specific or less specific route SHOULD combine them into the same UPDATE message.

Appendix F.3 Path attribute ordering

Implementations which combine update messages as described above in 6.1 may prefer to see all path attributes presented in a known order. This permits them to quickly identify sets of attributes from different update messages which are semantically identical. To facilitate this, it is a useful optimization to order the path attributes according to type code. This optimization is entirely optional.

Appendix F.4 AS_SET sorting

Another useful optimization that can be done to simplify this situation is to sort the AS numbers found in an AS_SET. This optimization is entirely optional.

Appendix F.5 Control over version negotiation

Since BGP-4 is capable of carrying aggregated routes which can not be properly represented in BGP-3, an implementation which supports BGP-4 and another BGP version should provide the capability to only speak BGP-4 on a per-peer basis.

Appendix F.6 Complex AS_PATH aggregation

An implementation which chooses to provide a path aggregation algorithm which retains significant amounts of path information may wish to use the following procedure:

For the purpose of aggregating AS_PATH attributes of two routes, we model each AS as a tuple <type, value>, where "type" identifies a type of the path segment the AS belongs to (e.g. AS_SEQUENCE, AS_SET), and "value" is the AS number. Two ASs are said to be the same if their corresponding <type, value> tuples are the same.

The algorithm to aggregate two AS_PATH attributes works as follows:

a) Identify the same ASs (as defined above) within each AS_PATH attribute that are in the same relative order within both AS_PATH attributes. Two ASs, X and Y, are said to be in the same order if either:
- X precedes Y in both AS_PATH attributes, or - Y precedes X in both AS_PATH attributes.

b) The aggregated AS_PATH attribute consists of ASs identified in (a) in exactly the same order as they appear in the AS_PATH attributes to be aggregated. If two consecutive ASs identified in (a) do not immediately follow each other in both of the AS_PATH attributes to be aggregated, then the intervening ASs (ASs that are between the two consecutive ASs that are the same) in both attributes are combined into an AS_SET path segment that consists of the intervening ASs from both AS_PATH

attributes; this segment is then placed in between the two consecutive ASs identified in (a) of the aggregated attribute. If two consecutive ASs identified in (a) immediately follow each other in one attribute, but do not follow in another, then the intervening ASs of the latter are combined into an AS_SET path segment; this segment is then placed in between the two consecutive ASs identified in (a) of the aggregated attribute.

c) For each pair of adjacent tuples in the aggregated AS_PATH, if both tuples have the same type, merge them together, as long as doing so will not cause a segment with length greater than 255 to be generated.

If as a result of the above procedure a given AS number appears more than once within the aggregated AS_PATH attribute, all, but the last instance (rightmost occurrence) of that AS number SHOULD be removed from the aggregated AS_PATH attribute.

Security Considerations

The authentication mechanism that an implementation of BGP MUST support is specified in [RFC2385]. The authentication provided by this mechanism could be done on a per peer basis.

BGP vulnerabilities analysis is discussed in [XXX].

IANA Considerations

All extensions to this protocol, including new message types and Path Attributes MUST only be made using the Standards Action process defined in [RFC2434].

Normative References

[RFC791] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification", RFC791, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC793, September 1981.

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

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC2385, August 1998.

[RFC2434] Narten, T., Alvestrand, H., "Guidelines for Writing an IANA Considerations Section in RFCs", RFC2434, October 1998

[RFC2474] Nichols, K., et al.,"Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC2474, December 1998

Non-normative References

[RFC904] Mills, D., "Exterior Gateway Protocol Formal Specification", RFC904, April 1984.

[RFC1092] Rekhter, Y., "EGP and Policy Based Routing in the New NSFNET Backbone", RFC1092, February 1989.

[RFC1093] Braun, H-W., "The NSFNET Routing Architecture", RFC1093, February 1989.

[RFC1772] Rekhter, Y., and P. Gross, "Application of the Border Gateway Protocol in the Internet", RFC1772, March 1995.

[RFC1518] Rekhter, Y., Li, T., "An Architecture for IP Address Allocation with CIDR", RFC 1518, September 1993.

[RFC1519] Fuller, V., Li, T., Yu, J., and Varadhan, K., ""Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC1519, September 1993.

[RFC1997] R. Chandra, P. Traina, T. Li, "BGP Communities Attribute", RFC 1997, August 1996.

[RFC2439] C. Villamizar, R. Chandra, R. Govindan, "BGP Route Flap Damping", RFC2439, November 1998.

[RFC2796] Bates, T., Chandra, R., Chen, E., "BGP Route Reflection - An Alternative to Full Mesh IBGP", RFC2796, April 2000.

[RFC2842] R. Chandra, J. Scudder, "Capabilities Advertisement with BGP-4", RFC2842.

[RFC2858] T. Bates, R. Chandra, D. Katz, Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC2858.

[RFC2918] Chen, E., "Route Refresh Capability for BGP-4", RFC2918, September 2000.

[RFC3065] Traina, P, McPherson, D., Scudder, J., "Autonomous System Confederations for BGP", RFC3065, February 2001.

[IS10747] "Information Processing Systems - Telecommunications and Information Exchange between Systems - Protocol for Exchange of Inter-domain Routeing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs", ISO/IEC IS10747, 1993

[XXX] Murphy, S., "BGP Security Vulnerabilities Analysis", draft-ietf-idr-bgp-vuln-00.txt, work in progress

Editors' Addresses

Yakov Rekhter
Juniper Networks
email: yakov@juniper.net

Tony Li
Procket Networks, Inc.
email: tli@procket.com

Susan Hares

NextHop Technologies, Inc.
email: skh@nexthop.com