Network Working Group
Request for Comments: 1163
Obsoletes: RFC 1105

K. Lougheed
cisco Systems
Y. Rekhter
T.J. Watson Research Center, IBM Corp
June 1990

A Border Gateway Protocol (BGP)

Status of this Memo

This RFC, together with its companion RFC-1164, "Application of the Border Gateway Protocol in the Internet", define a Proposed Standard for an inter-autonomous system routing protocol for the Internet.

This protocol, like any other at this initial stage, may undergo modifications before reaching full Internet Standard status as a result of deployment experience. Implementers are encouraged to track the progress of this or any protocol as it moves through the standardization process, and to report their own experience with the protocol.

This protocol is being considered by the Interconnectivity Working Group (IWG) of the Internet Engineering Task Force (IETF). Information about the progress of BGP can be monitored and/or reported on the IWG mailing list (IWG@nri.reston.va.us).

Please refer to the latest edition of the "IAB Official Protocol Standards" RFC for current information on the state and status of standard Internet protocols.

Distribution of this memo is unlimited.

Table of Contents

1. Acknowledgements

We would like to express our thanks to Guy Almes (Rice University), Len Bosack (cisco Systems), Jeffrey C. Honig (Cornell Theory Center) and all members of the Interconnectivity Working Group of the Internet Engineering Task Force, chaired by Guy Almes, for their contributions to this document.

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 earlier versions 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.

2. Introduction

The Border Gateway Protocol (BGP) is an inter-Autonomous System routing protocol. It is built on experience gained with EGP as defined in RFC 904 [1] and EGP usage in the NSFNET Backbone as described in RFC 1092 [2] and RFC 1093 [3].

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 full path of Autonomous Systems (ASs) that traffic must transit to reach these networks. 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.

To characterize the set of policy decisions that can be enforced using BGP, one must focus on the rule that an AS advertize to its

neighbor ASs only those routes that it itself uses. This rule reflects the "hop-by-hop" routing paradigm generally used throughout the current Internet. Note that some policies cannot be supported by the "hop-by-hop" routing paradigm and thus require techniques such as source routing to enforce. For example, BGP does not enable one AS to send traffic to a neighbor AS intending that that traffic take a different route from that taken by traffic originating in the neighbor AS. On the other hand, BGP can support any policy conforming to the "hop-by-hop" routing paradigm. Since the current Internet uses only the "hop-by-hop" routing paradigm and since BGP can support any policy that conforms to that paradigm, BGP is highly applicable as an inter-AS routing protocol for the current Internet.

A more complete discussion of what policies can and cannot be enforced with BGP is outside the scope of this document (but refer to the companion document discussing BGP usage [5]).

BGP runs over a reliable transport protocol. This eliminates the need to implement explicit update fragmentation, retransmission, acknowledgement, and sequencing. Any authentication scheme used by the transport protocol may be used in addition to BGP's own authentication mechanisms. The error notification mechanism used in BGP assumes that the transport protocol supports a "graceful" close, i.e., that all outstanding data will be delivered before the connection is closed.

BGP uses TCP [4] as its transport protocol. TCP meets BGP's transport requirements and is present in virtually all commercial routers and hosts. In the following descriptions the phrase "transport protocol connection" can be understood to refer to a TCP connection. BGP uses TCP port 179 for establishing its connections.

This memo 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 and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other ASs. Since this classic definition was developed, it has become common for a single AS to use several interior gateway protocols 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 networks are reachable through it. From the standpoint of exterior routing, an AS can be viewed as monolithic: reachability to networks directly connected to the AS must be equivalent from all border gateways of the AS.

The planned use of BGP in the Internet environment, including such issues as topology, the interaction between BGP and IGPs, and the enforcement of routing policy rules is presented in a companion document [5]. This document is the first of a series of documents planned to explore various aspects of BGP application.

3. Summary of Operation

Two systems form a transport protocol connection between one another. They exchange messages to open and confirm the connection parameters. The initial data flow is the entire BGP routing table. Incremental updates are sent as the routing tables change. BGP does not require periodic refresh of the entire BGP routing table. Therefore, a BGP speaker must retain the current version of the entire BGP routing tables of all of its peers for the duration of the connection. KeepAlive messages are 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.

The hosts executing the Border Gateway Protocol need not be routers. A non-routing host could exchange routing information with routers via EGP or even an interior routing protocol. That non-routing host could then use BGP to exchange routing information with a border router in another Autonomous System. The implications and applications of this architecture are for further study.

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 interior routing protocol. A consistent view of the routes exterior to the AS can be provided by having all BGP speakers within the AS maintain direct BGP connections with each other. Using a common set of policies, the BGP speakers arrive at an agreement as to which border routers will serve as exit/entry points for particular networks outside the AS. This information is communicated to the AS's internal routers, possibly via the interior routing protocol. 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.

Connections between BGP speakers of different ASs are referred to as "external" links. BGP connections between BGP speakers within the same AS are referred to as "internal" links.

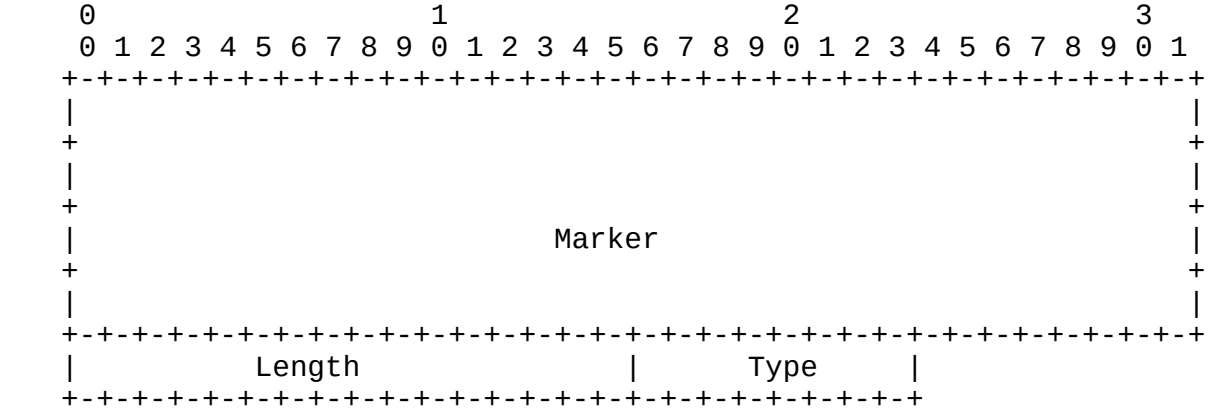
4. Message Formats

This section describes message formats used by BGP.

Messages are sent over a reliable transport protocol 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.

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 contains a value that the receiver of the message can predict. If the Type of the message is OPEN, or if the Authentication Code used in the OPEN message of the connection is zero, then the Marker must be all ones. Otherwise, the value of the marker can be predicted by some a computation specified as part of the authentication mechanism used. The Marker can be used to detect loss of synchronization between a pair of BGP peers, and to authenticate incoming BGP messages.

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 transport-level 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:

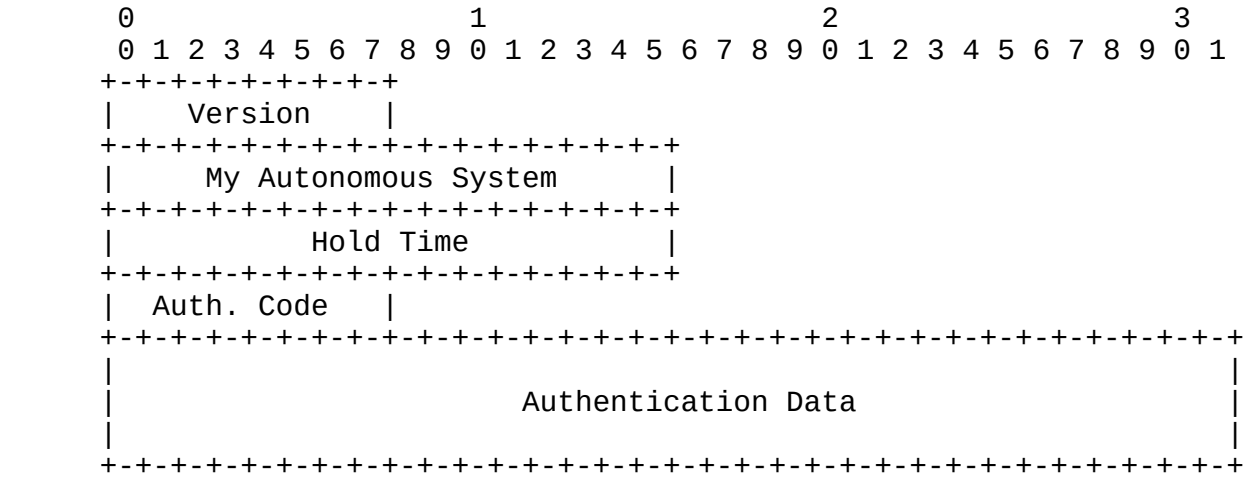
This 1-octet unsigned integer indicates the type code of the message. The following type codes are defined:

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

4.2 OPEN Message Format

After a transport protocol connection 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. Once the OPEN is confirmed, UPDATE, KEEPALIVE, and NOTIFICATION messages may be exchanged.

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 2.

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 maximum number of seconds that may elapse between the receipt of successive KEEPALIVE and/or UPDATE and/or NOTIFICATION messages.

Authentication Code:

This 1-octet unsigned integer indicates the authentication mechanism being used. Whenever an authentication mechanism is specified for use within BGP, three things must be included in the specification:

- the value of the Authentication Code which indicates use of the mechanism,
- the form and meaning of the Authentication Data, and
- the algorithm for computing values of Marker fields.

Only one authentication mechanism is specified as part of this memo:

- its Authentication Code is zero,
- its Authentication Data must be empty (of zero length), and
- the Marker fields of all messages must be all ones.

The semantics of non-zero Authentication Codes lies outside the scope of this memo.

Note that a separate authentication mechanism may be used in establishing the transport level connection.

Authentication Data:

The form and meaning of this field is a variable-length field depend on the Authentication Code. If the value of Authentication Code field is zero, the Authentication Data field must have zero length. The semantics of the non-zero length Authentication Data field is outside the scope of this memo.

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

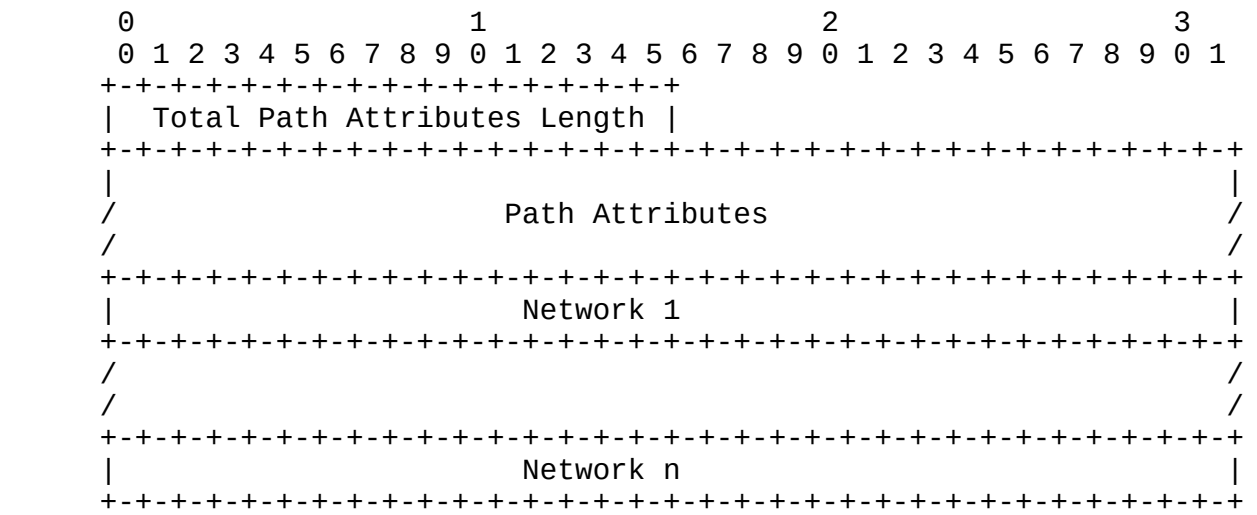
Message Length = 25 + Authentication Data Length

The minimum length of the OPEN message is 25 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 packet 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.

In addition to the fixed-size BGP header, the UPDATE message contains the following fields (note that all fields may have arbitrary alignment):



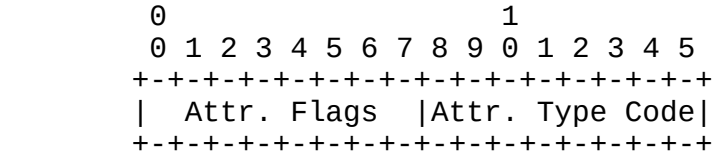
Total Path Attribute Length:

This 2-octet unsigned integer indicates the total length of the Path Attributes field in octets. Its value must allow the (non-negative integer) number of Network fields to be determined as specified below.

Path Attributes:

A variable length sequence of path attributes is present in every UPDATE. 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). Extended Length may be used only if the length of the attribute value is greater than 255 octets.

The lower-order four bits of the Attribute Flags octet are unused. They must be zero (and should 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 meaning and handling of Path Attributes is discussed in

Section 5.

Network:

Each 4-octet Internet network number indicates one network whose Inter-Autonomous System routing is described by the Path Attributes. Subnets and host addresses are specifically not allowed. The total number of Network fields in the UPDATE message can be determined by the formula:

Message Length = 19 + Total Path Attribute Length + 4 * #Nets

The message Length field of the message header and the Path Attributes Length field of the UPDATE message must be such that the formula results in a non-negative integer number of Network fields.

The minimum length of the UPDATE message is 37 octets (including message header).

4.4 KEEPALIVE Message Format

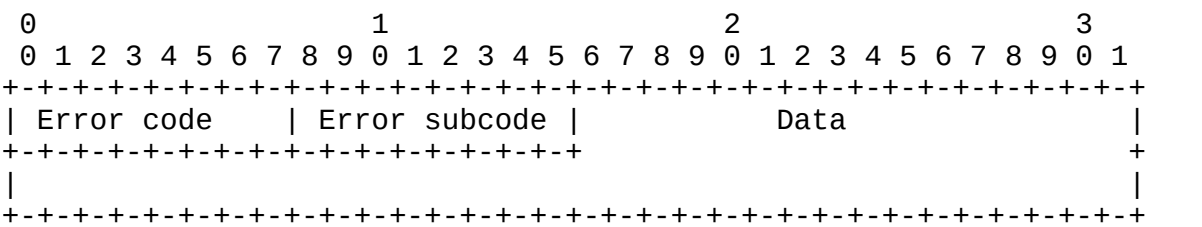
BGP does not use any transport protocol-based keep-alive mechanism to determine if peers are reachable. Instead, KEEPALIVE messages are exchanged between peers often enough as not to cause the hold time (as advertised in the OPEN message) to expire. A reasonable maximum time between KEEPALIVE messages would be one third of the Hold Time interval.

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 - Unsupported Authentication Code.
4 - Authentication Failure.

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 - AS Routing Loop.
8 - Invalid NEXT_HOP Attribute.
9 - Optional Attribute Error.
10 - Invalid Network Field.

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. Others are discretionary and may or may not be sent in a particular UPDATE message. Which well-known attributes are mandatory or discretionary is noted in the table below.

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. Unrecognized transitive optional attributes should be accepted and passed along to other BGP peers. If a path with unrecognized transitive optional attribute is accepted and passed along to other BGP peers, the Partial bit in the Attribute Flags octet is 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 should

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 AS 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 INTER-AS METRIC attribute gives an example.) All optional attributes (both transitive and non-transitive) may be updated (if appropriate) by ASs in the path.

The order of attributes within the Path Attributes field of a particular UPDATE message is irrelevant.

The same attribute cannot appear more than once within the Path Attributes field of a particular UPDATE message.

Following table specifies attribute type code, attribute length, and attribute category for path attributes defined in this document:

Attribute Name	Type Code	Length	Attribute category
ORIGIN	1	1	well-known, mandatory
AS_PATH	2	variable	well-known, mandatory
NEXT_HOP	3	4	well-known, mandatory
UNREACHABLE	4	0	well-known, discretionary
INTER-AS METRIC	5	2	optional, non-transitive

ORIGIN:

The ORIGIN path attribute defines the origin of the path information. The data octet can assume the following values:

Value	Meaning
0	IGP - network(s) are interior to the originating AS
1	EGP - network(s) learned via EGP
2	INCOMPLETE - network(s) learned by some other means

AS_PATH:

The AS_PATH attribute enumerates the ASs that must be traversed to reach the networks listed in the UPDATE message. Since an AS identifier is 2 octets, the length of an AS_PATH attribute is twice the number of ASs in the path. Rules for constructing an AS_PATH attribute are discussed in Section 9.

NEXT_HOP:

The NEXT_HOP path attribute defines the IP address of the border router that should be used as the next hop to the networks listed in the UPDATE message. This border router must belong to the same AS as the BGP peer that advertises it.

UNREACHABLE:

The UNREACHABLE attribute is used to notify a BGP peer that some of the previously advertised routes have become unreachable.

INTER-AS METRIC:

The INTER-AS METRIC attribute may be used on external (inter-AS) links to discriminate between multiple exit or entry points to the same neighboring AS. The value of the INTER-AS METRIC attribute is a 2-octet unsigned number which is called a metric. All other factors being equal, the exit or entry point with lower metric should be preferred. If received over external links, the INTER-AS METRIC attribute may be propagated over internal links to other BGP speaker within the same AS. The INTER-AS METRIC attribute is never propagated to other BGP speakers in neighboring AS's.

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. If no Error Subcode is specified, then a zero should be used.

The phrase "the BGP connection is closed" means that the transport protocol connection has been closed and that all resources for that BGP connection have been deallocated. Routing table entries 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 message type is OPEN. The expected value of the Marker field for all other types of BGP messages determined based on the Authentication Code in the BGP OPEN message and the actual authentication mechanism (if the Authentication Code in the BGP OPEN message is non-zero). If the Marker field of the message header is not the expected one, 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-octet 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).

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 Authentication Code of the OPEN message is not recognized, then the Error Subcode is set to Unsupported Authentication Code.

If the Authentication Code is zero, then the Authentication Data must be of zero length. Otherwise, the Error Subcode is set to Authentication Failure.

If the Authentication Code is non-zero, then the corresponding authentication procedure is invoked. If the authentication procedure (based on Authentication Code and Authentication Data) fails, then the Error Subcode is set to Authentication Failure.

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 Total Attribute Length is too large (i.e., if Total Attribute Length + 21 exceeds the message Length), or if the (non-negative integer) Number of Network fields cannot be computed as in Section 4.3, 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 AS route specified by the AS_PATH attribute is checked for AS loops. AS loop detection is done by scanning the full AS route (as specified in the AS_PATH attribute) and checking that each AS occurs at most once. If a loop is detected, then the Error Subcode is set to AS Routing Loop. The Data field contains the incorrect attribute (type, length and value).

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.

Each Network field in the UPDATE message is checked for syntactic validity. If the Network field is syntactically incorrect, or contains a subnet or a host address, then the Error Subcode is set to Invalid Network Field.

6.4 NOTIFICATION message error handling.

If a peer sends a NOTIFICATION message, and there is an error in that message, there is unfortunately no means of reporting this error via a subsequent NOTIFICATION message. 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 should be sent and the BGP connection 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 should not be used when a fatal error indicated by this section does exist.

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 BGP operation in terms of a Finite State Machine (FSM). Following is a brief summary and overview of BGP operations by state as determined by this FSM. A condensed version of the BGP FSM is found in Appendix 1.

Initially BGP is in the Idle state.

Idle state:

In this state BGP refuses all incoming BGP connections. No resources are allocated to the BGP neighbor. In response to the Start event (initiated by either system or operator) the local system initializes all BGP resources, starts the ConnectRetry timer, initiates a transport connection to other BGP peer, while listening for connection that may be initiated by the remote BGP peer, and changes its state to Connect.

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

Any other event received in the Idle state is ignored.

Connect state:

In this state BGP is waiting for the transport protocol connection to be completed.

If the transport protocol connection succeeds, the local system clears the ConnectRetry timer, completes initialization, sends an OPEN message to its peer, and changes its state to OpenSent.

If the transport protocol connect fails (e.g., retransmission timeout), the local system restarts the ConnectRetry timer, continues to listen for a connection that may be initiated by the remote BGP peer, and changes its state to Active state.

In response to the ConnectRetry timer expired event, the local system restarts the ConnectRetry timer, initiates a transport connection to other BGP peer, continues to listen for a connection that may be initiated by the remote BGP peer, and stays in the Connect state.

Start event is ignored in the Active state.

In response to any other event (initiated by either system or operator), the local system releases all BGP resources associated with this connection and changes its state to Idle.

Active state:

In this state BGP is trying to acquire a BGP neighbor by initiating a transport protocol connection.

If the transport protocol connection succeeds, the local system clears the ConnectRetry timer, completes initialization, sends an OPEN message to its peer, sets its hold timer to a large value, and changes its state to OpenSent.

In response to the ConnectRetry timer expired event, the local system restarts the ConnectRetry timer, initiates a transport connection to other BGP peer, continues to listen for a connection that may be be initiated by the remote BGP peer, and changes its state to Connect.

If the local system detects that a remote peer is trying to

establish BGP connection to it, and the IP address of the remote peer is not an expected one, the local system restarts the ConnectRetry timer, rejects the attempted connection, continues to listen for a connection that may be initiated by the remote BGP peer, and stays in the Active state.

Start event is ignored in the Active state.

In response to any other event (initiated by either system or operator), the local system releases all BGP resources associated with this connection and changes its state to Idle.

OpenSent state:

In this state BGP waits for an OPEN message from its peer. When an OPEN message is received, all fields are checked for correctness. If the BGP message header checking or OPEN message checking detects an error (see Section 6), the local system sends a NOTIFICATION message and changes its state to Idle.

If there are no errors in the OPEN message, BGP sends a KEEPALIVE message and sets a KeepAlive timer. The hold timer, which was originally set to an arbitrary large value (see above), is replaced with the value indicated in the OPEN message. If the value of the Autonomous System field is the same as our own , then the connection is "internal" connection; otherwise, it is "external". (This will effect UPDATE processing as described below.) Finally, the state is changed to OpenConfirm.

If a disconnect notification is received from the underlying transport protocol, the local system closes the BGP connection, restarts the ConnectRetry timer, while continue listening for connection that may be initiated by the remote BGP peer, and goes into the Active state.

If the hold time expires, the local system sends NOTIFICATION message with error code Hold Timer Expired and changes its state to Idle.

In response to the Stop event (initiated by either system or operator) the local system sends NOTIFICATION message with Error Code Cease and changes its state to Idle.

Start event is ignored in the OpenSent state.

In response to any other event the local system sends

NOTIFICATION message with Error Code Finite State Machine Error and changes its state to Idle.

Whenever BGP changes its state from OpenSent to Idle, it closes the BGP (and transport-level) connection and releases all resources associated with that connection.

OpenConfirm state:

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

If the local system receives a KEEPALIVE message, it changes its state to Established.

If the hold timer expires before a KEEPALIVE message is received, the local system sends NOTIFICATION message with error code Hold Timer expired and changes its state to Idle.

If the local system receives a NOTIFICATION message, it changes its state to Idle.

If the KeepAlive timer expires, the local system sends a KEEPALIVE message and restarts its KeepAlive timer.

If a disconnect notification is received from the underlying transport protocol, the local system changes its state to Idle.

In response to the Stop event (initiated by either system or operator) the local system sends NOTIFICATION message with Error Code Cease and changes its state to Idle.

Start event is ignored in the OpenConfirm state.

In response to any other event the local system sends NOTIFICATION message with Error Code Finite State Machine Error and changes its state to Idle.

Whenever BGP changes its state from OpenConfirm to Idle, it closes the BGP (and transport-level) connection and releases all resources associated with that connection.

Established state:

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

If the local system receives an UPDATE or KEEPALIVE message, it

restarts its Holdtime timer.

If the local system receives a NOTIFICATION message, it changes its state to Idle.

If the local system receives an UPDATE message and the UPDATE message error handling procedure (see Section 6.3) detects an error, the local system sends a NOTIFICATION message and changes its state to Idle.

If a disconnect notification is received from the underlying transport protocol, the local system changes its state to Idle.

If the Holdtime timer expires, the local system sends a NOTIFICATION message with Error Code Hold Timer Expired and changes its state to Idle.

If the KeepAlive timer expires, the local system sends a KEEPALIVE message and restarts its KeepAlive timer.

Each time the local system sends a KEEPALIVE or UPDATE message, it restarts its KeepAlive timer.

In response to the Stop event (initiated by either system or operator), the local system sends a NOTIFICATION message with Error Code Cease and changes its state to Idle.

Start event is ignored in the Established state.

In response to any other event, the local system sends NOTIFICATION message with Error Code Finite State Machine Error and changes its state to Idle.

Whenever BGP changes its state from Established to Idle, it closes the BGP (and transport-level) connection, releases all resources associated with that connection, and deletes all routes derived from that connection.

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 network and the path attributes associated with a route to that network are correct, then the route is compared with other routes to the same network. If the new route is better than the current one, then it is propagated via an UPDATE message to adjacent BGP speakers as follows:

- If a route in the UPDATE was received over an internal link, it is not propagated over any other internal link. This restriction is due to the fact that all BGP speakers within a single AS form a completely connected graph (see above).

- If the UPDATE message is propagated over an external link, then the local AS number is prepended to the AS_PATH attribute, and the NEXT_HOP attribute is updated with an IP address of the router that should be used as a next hop to the network. If the UPDATE message is propagated over an internal link, then the AS_PATH attribute is passed unmodified and the NEXT_HOP attribute is replaced with the sender's own IP address.

Generally speaking, the 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 cannot be viewed as better than any other route. If such a route were ever used, a routing loop would result.

- In order to achieve successful distributed operation, only routes with a likelihood of stability can be chosen. Thus, an AS must avoid using unstable routes, and it must 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.

10. Detection of Inter-AS Policy Contradictions

Since BGP requires no central authority for coordinating routing policies among ASs, and since routing policies are not exchanged via the protocol itself, it is possible for a group of ASs to have a set

of routing policies that cannot simultaneously be satisfied. This may cause an indefinite oscillation of the routes in this group of ASs.

To help detect such a situation, all BGP speakers must observe the following rule. If a route to a destination that is currently used by the local system is determined to be unreachable (e.g., as a result of receiving an UPDATE message for this route with the UNREACHABLE attribute), then, before switching to another route, this local system must advertize this route as unreachable to all the BGP neighbors to which it previously advertized this route.

This rule will allow other ASs to distinguish between two different situations:

- The local system has chosen to use a new route because the old route become unreachable.

- The local system has chosen to use a new route because it preferred it over the old route. The old route is still viable.

In the former case, an UPDATE message with the UNREACHABLE attribute will be received for the old route. In the latter case it will not.

In some cases, this may allow a BGP speaker to detect the fact that its policies, taken together with the policies of some other AS, cannot simultaneously be satisfied. For example, consider the following situation involving AS A and its neighbor AS B. B advertises a route with a path of the form <B,...>, where A is not present in the path. A then decides to use this path, and advertises <A,B,...> to all its neighbors. B later advertises <B,...,A,...> back to A, without ever declaring its previous path <B,...> to be unreachable. Evidently, A prefers routes via B and B prefers routes via A. The combined policies of A and B, taken together, cannot be satisfied. Such an event should be noticed, logged locally, and brought to the attention of AS A's administration. The means to do this, however, lies outside the scope of this document. Also outside the document is a more complete procedure for detecting such contradictions of policy.

While the above rules provide a mechanism to detect a set of routing policies that cannot be satisfied simultaneously, the protocol itself does not provide any mechanism for suppressing the route oscillation that may result from these unsatisfiable policies. The reason for doing this is that routing policies are viewed as external to the protocol and as determined by the local AS administrator.

Appendix 1. BGP FSM State Transitions and Actions.

This Appendix discusses the transitions between states in the BGP FSM in response to BGP events. The following is the list of these states and events.

BGP States:

1 - Idle
2 - Connect
3 - Active
4 - OpenSent
5 - OpenConfirm
6 - Established

BGP Events:

1 - BGP Start
2 - BGP Stop
3 - BGP Transport connection open
4 - BGP Transport connection closed
5 - BGP Transport connection open failed
6 - BGP Transport fatal error
7 - ConnectRetry timer expired
8 - Holdtime timer expired
9 - KeepAlive timer expired
10 - Receive OPEN message
11 - Receive KEEPALIVE message
12 - Receive UPDATE messages
13 - Receive NOTIFICATION message

The following table describes the state transitions of the BGP FSM and the actions triggered by these transitions.

Event	Actions	Message Sent	Next State
Idle (1)			
1	Initialize resources Start ConnectRetry timer Initiate a transport connection	none	2
others	none	none	1

Connect(2)			
1	none	none	2
3	Complete initialization Clear ConnectRetry timer	OPEN	4
5	Restart ConnectRetry timer	none	3
7	Restart ConnectRetry timer Initiate a transport connection	none	2
others	Release resources	none	1
Active (3)			
1	none	none	3
3	Complete initialization Clear ConnectRetry timer	OPEN	4
5	Close connection Restart ConnectRetry timer		3
7	Restart ConnectRetry timer Initiate a transport connection	none	2
others	Release resources	none	1
OpenSent(4)			
1	none	none	4
4	Close transport connection Restart ConnectRetry timer	none	3
6	Release resources	none	1
10	Process OPEN is OK	KEEPALIVE	5
	Process OPEN failed	NOTIFICATION	1
others	Close transport connection Release resources	NOTIFICATION	1
OpenConfirm (5)			
1	none	none	5
4	Release resources	none	1
6	Release resources	none	1
9	Restart KeepAlive timer	KEEPALIVE	5
11	Complete initialization Restart Holdtime timer	none	6
13	Close transport connection Release resources		1
others	Close transport connection Release resources	NOTIFICATION	1

Established (6)			
1	none	none	6
4	Release resources	none	1
6	Release resources	none	1
9	Restart KeepAlive timer	KEEPALIVE	6
11	Restart Holdtime timer	KEEPALIVE	6
12	Process UPDATE is OK	UPDATE	6
	Process UPDATE failed	NOTIFICATION	1
13	Close transport connection Release resources		1
others	Close transport connection Release resources	NOTIFICATION	1

The following is a condensed version of the above state transition table.

Events	Idle (1)	Active (2)	Connect (3)	OpenSent (4)	OpenConfirm (5)	Estab (6)
1	2	2	3	4	5	6
2	1	1	1	1	1	1
3	1	4	4	1	1	1
4	1	1	1	3	1	1
5	1	3	3	1	1	1
6	1	1	1	1	1	1
7	1	2	2	1	1	1
8	1	1	1	1	1	1
9	1	1	1	1	5	6
10	1	1	1	1 or 5	1	1
11	1	1	1	1	6	6
12	1	1	1	1	1	1 or 6
13	1	1	1	1	1	1

Appendix 2. Comparison with RFC 1105

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.

Appendix 3. 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 precedence for TCP connection, then the BGP transport connection should be opened with precedence set to Internetwork Control (110) value (see also [6]).

References

[1] Mills, D., "Exterior Gateway Protocol Formal Specification", RFC 904, BBN, April 1984.

[2] Rekhter, Y., "EGP and Policy Based Routing in the New NSFNET Backbone", RFC 1092, T.J. Watson Research Center, February 1989.

[3] Braun, H-W., "The NSFNET Routing Architecture", RFC 1093, MERIT/NSFNET Project, February 1989.

[4] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[5] Honig, J., Katz, D., Mathis, M., Rekhter, Y., and J. Yu, "Application of the Border Gateway Protocol in the Internet", RFC 1164, Cornell University Theory Center, Merit/NSFNET, Pittsburgh Supercomputing Center, IBM, Merit/NSFNET, June 1990.

[6] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, DARPA, September 1981.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Kirk Lougheed
cisco Systems, Inc.
1525 O'Brien Drive
Menlo Park, CA 94025

Phone: (415) 326-1941

Email: LOUGHEED@CISCO.COM

Yakov Rekhter
T.J. Watson Research Center IBM Corporation
P.O. Box 218
Yorktown Heights, NY 10598

Phone: (914) 945-3896

Email: YAKOV@IBM.COM