Network Working Group
Request for Comments: 3913
Category: Informational

D. Thaler
Microsoft

September 2004

Border Gateway Multicast Protocol (BGMP): Protocol Specification

Status of this Memo

Copyright Notice

Abstract

This document describes the Border Gateway Multicast Protocol (BGMP), a protocol for inter-domain multicast routing. BGMP builds shared trees for active multicast groups, and optionally allows receiver domains to build source-specific, inter-domain, distribution branches where needed. BGMP natively supports "source-specific multicast" (SSM). To also support "any-source multicast" (ASM), BGMP requires that each multicast group be associated with a single root (in BGMP it is referred to as the root domain). It requires that different ranges of the multicast address space are associated (e.g., with Unicast-Prefix-Based Multicast addressing) with different domains. Each of these domains then becomes the root of the shared domain-trees for all groups in its range. Multicast participants will generally receive better multicast service if the session initiator's address allocator selects addresses from its own domain's part of the space, thereby causing the root domain to be local to at least one of the session participants.

Table of Contents

1. Purpose

It has been suggested that inter-domain "any-source" multicast is better supported with a rendezvous mechanism whereby members receive sources' data packets without any sort of global broadcast (e.g., MSDP broadcasts source information, PIM-DM [PIMDM] and DVMRP [DVMRP] broadcast initial data packets, and MOSPF [MOSPF] broadcasts membership information). PIM-SM [PIMSM] and CBT [CBT] use a shared group-tree, to which all members join and thereby hear from all sources (and to which non-members do not join and thereby hear from no sources).

This document describes BGMP, a protocol for inter-domain multicast routing. BGMP natively supports "source-specific multicast" (SSM). To also support "any-source multicast" (ASM), BGMP builds shared trees for active multicast groups, and allows domains to build source-specific, inter-domain, distribution branches where needed. Building upon concepts from PIM-SM and CBT, BGMP requires that each global multicast group be associated with a single root. However, in BGMP, the root is an entire exchange or domain, rather than a single router.

For non-source-specific groups, BGMP assumes that ranges of the multicast address space have been associated (e.g., with Unicast-Prefix-Based Multicast [V4PREFIX,V6PREFIX] addressing) with selected domains. Each such domain then becomes the root of the shared domain-trees for all groups in its range. An address allocator will generally achieve better distribution trees if it takes its multicast addresses from its own domain's part of the space, thereby causing the root domain to be local.

BGMP uses TCP as its transport protocol. This eliminates the need to implement message fragmentation, retransmission, acknowledgement, and sequencing. BGMP uses TCP port 264 for establishing its connections. This port is distinct from BGP's port to provide protocol independence, and to facilitate distinguishing between protocol packets (e.g., by packet classifiers, diagnostic utilities, etc.)

Two BGMP peers form a TCP connection between one another, and exchange messages to open and confirm the connection parameters. They then send incremental Join/Prune Updates as group memberships change. BGMP does not require periodic refresh of individual entries. 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 if the error is a fatal one.

2. Terminology

This document uses the following technical terms:

Domain:
A set of one or more contiguous links and zero or more routers surrounded by one or more multicast border routers. Note that this loose definition of domain also applies to an external link between two domains, as well as an exchange.

Root Domain:
When constructing a shared tree of domains for some group, one domain will be the "root" of the tree. The root domain receives data from each sender to the group, and functions as a rendezvous domain toward which member domains can send inter-domain joins, and to which sender domains can send data.

Multicast RIB:
The Routing Information Base, or routing table, used to calculate the "next-hop" towards a particular address for multicast traffic.

Multicast IGP (M-IGP):
A generic term for any multicast routing protocol used for tree construction within a domain. Typical examples of M-IGPs are: PIM-SM, PIM-DM, DVMRP, MOSPF, and CBT.

EGP: A generic term for the interdomain unicast routing protocol in use.
Typically, this will be some version of BGP which can support a Multicast RIB, such as MBGP [MBGP], containing both unicast and multicast address prefixes.

Component:
The portion of a border router associated with (and logically inside) a particular domain that runs the multicast IGP (M-IGP) for that domain, if any. Each border router thus has zero or more components inside routing domains. In addition, each border router with external links that do not fall inside any routing domain will have an inter-domain component that runs BGMP.

External peer:
A border router in another multicast AS (autonomous system, as used in BGP), to which a BGMP TCP-connection is open. If BGP is being used as the EGP, a separate "eBGP" TCP-connection will also be open to the same peer.

Internal peer:
Another border router of the same multicast AS. If BGP is being used as the EGP, the border router either speaks iBGP ("internal" BGP) directly to internal peers in a full mesh, or indirectly through a route reflector [REFLECT].

Next-hop peer:
The next-hop peer towards a given IP address is the next EGP router on the path to the given address, according to multicast RIB routes in the EGP's routing table (e.g., in MBGP, routes whose Subsequent Address Family Identifier field indicates that the route is valid for multicast traffic).

target:
Either an EGP peer, or an M-IGP component.

Tree State Table:
This is a table of (S-prefix,G) and (*,G-prefix) entries that have been explicitly joined by a set of targets. Each entry has, in addition to the source and group addresses and masks, a list of targets that have explicitly requested data (on behalf of directly connected hosts or downstream routers). (S,G) entries also have an "SPT" bit.

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as described in [RFC2119].

3. Protocol Overview

BGMP maintains group-prefix state in response to messages from BGMP peers and notifications from M-IGP components. Group-shared trees are rooted at the domain advertising the group prefix covering those groups. When a receiver joins a specific group address, the border router towards the root domain generates a group-specific Join message, which is then forwarded Border-Router-by-Border-Router towards the root domain (see Figure 1). BGMP Join and Prune messages are sent over TCP connections between BGMP peers, and BGMP protocol state is refreshed by KEEPALIVE messages periodically sent over TCP.

BGMP routers build group-specific bidirectional forwarding state as they process the BGMP Join messages. Bidirectional forwarding state means that packets received from any target are forwarded to all other targets in the target list without any RPF checks. No group-specific state or traffic exists in parts of the network where there are no members of that group.

BGMP routers optionally build source-specific unidirectional forwarding state, only where needed, to be compatible with source-specific trees (SPTs) used by some M-IGPs (e.g., DVMRP, PIM-DM, or PIM-SM), or to construct trees for source-specific groups. A domain that uses an SPT-based M-IGP may need to inject multicast packets from external sources via different border routers (to be compatible with the M-IGP RPF checks) which thus act as "surrogates". For example, in the Transit_1 domain, data from Src_A arrives at BR12, but must be injected by BR11. A surrogate router may create a source-specific BGMP branch if no shared tree state exists. Note: stub domains with a single border router, such as Rcvr_Stub_7 in Figure 1, receive all multicast data packets through that router, to which all RPF checks point. Therefore, stub domains never build source-specific state.

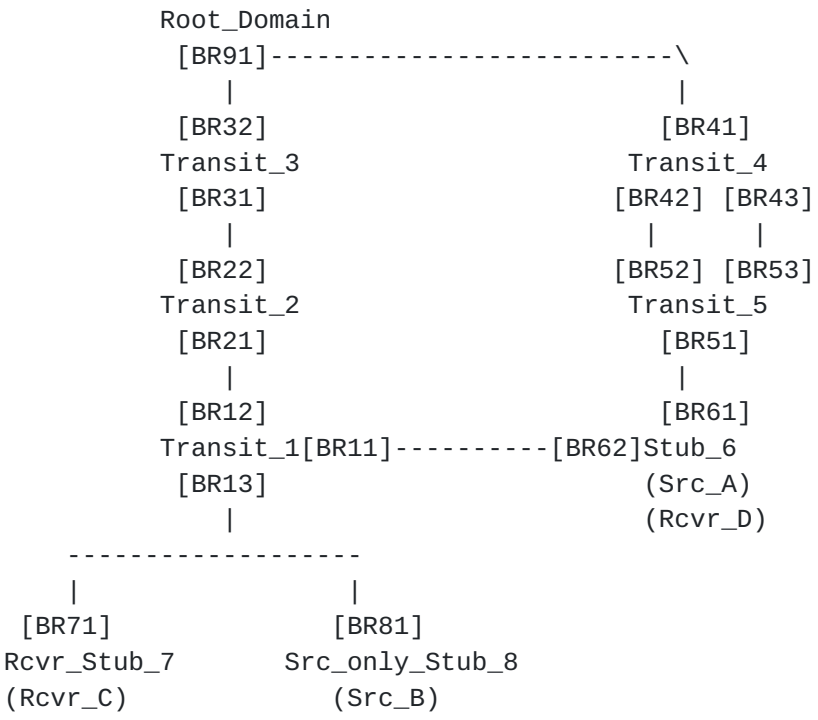


Figure 1: Example inter-domain topology. [BRxy] represents a BGMP border router. Transit_X is a transit domain network. *_Stub_X is a stub domain network.

Data packets are forwarded based on a combination of BGMP and M-IGP rules. The router forwards to a set of targets according to a matching (S,G) BGMP tree state entry if it exists. If not found, the router checks for a matching (*,G) BGMP tree state entry. If neither is found, then the packet is sent natively to the next-hop EGP peer for G, according to the Multicast RIB (for example, in the case of a non-member sender such as Src_B in Figure 1). If a matching entry was found, the packet is forwarded to all other targets in the target

list. In this way BGMP trees forward data in a bidirectional manner. If a target is an M-IGP component then forwarding is subject to the rules of that M-IGP protocol.

3.1. Design Rationale

Several other protocols, or protocol proposals, build shared trees within domains [PIMSM, CBT]. The design choices made for BGMP result from our focus on Inter-Domain multicast in particular. The design choices made by PIM-SM and CBT are better suited to the wide-area intra-domain case. There are three major differences between BGMP and other shared-tree protocols:

(1) Unidirectional vs. Bidirectional trees

Bidirectional trees (using bidirectional forwarding state as described above) minimize third party dependence which is essential in the inter-domain context. For example, in Figure 1, stub domains 7 and 8 would like to exchange multicast packets without being dependent on the quality of connectivity of the root domain. However, unidirectional shared trees (i.e., those using RPF checks) have more aggressive loop prevention and share the same processing rules as source-specific entries which are inherently unidirectional.

The lack of third party dependence concerns in the INTRA domain case reduces the incentive to employ bidirectional trees. BGMP supports bidirectional trees because it has to, and because it can without excessive cost.

(2) Source-specific distribution trees/branches

In a departure from other shared tree protocols, source-specific BGMP state is built ONLY where (a) it is needed to pull the multicast traffic down to a BGMP router that has source-specific (S,G) state, and (b) that router is NOT already on the shared tree (i.e., has no (*,G) state), and (c) that router does not want to receive packets via encapsulation from a router which is on the shared tree. BGMP provides source-specific branches because most M-IGP protocols in use today build source-specific trees. BGMP's source-specific branches eliminate the unnecessary overhead of encapsulations for high data rate sources from the shared tree's ingress router to the surrogate injector (e.g., from BR12 to BR11 in Figure 1). Moreover, cases in which shared paths are significantly longer than SPT paths will also benefit.

However, except for source-specific group distribution trees, we do not build source-specific inter-domain trees in general because (a) inter-domain connectivity is generally less rich than intra-domain

connectivity, so shared distribution trees should have more acceptable path length and traffic concentration properties in the inter-domain context, than in the intra-domain case, and (b) by having the shared tree state always take precedence over source-specific tree state, we avoid ambiguities that can otherwise arise.

In summary, BGMP trees are, in a sense, a hybrid between PIM-SM and CBT trees.

(3) Method of choosing root of group shared tree

The choice of a group's shared-tree-root has implications for performance and policy. In the intra-domain case it is sometimes assumed that all potential shared-tree roots (RPs/Cores) within the domain are equally suited to be the root for a group that is initiated within that domain. In the INTER-domain case, there is far more opportunity for unacceptably poor locality, and administrative control of a group's shared-tree root. Therefore in the intra-domain case, other protocols sometimes treat all candidate roots (RPs or Cores) as equivalent and emphasize load sharing and stability to maximize performance. In the Inter-Domain case, all roots are not equivalent, and we adopt an approach whereby a group's root domain is not random but is subject to administrative control.

4. Protocol Details

In this section, we describe the detailed protocol that border routers perform. We assume that each border router conforms to the component-based model described in [INTEROP], modulo one correction to section 3.2 ("BGMP" Dispatcher), as follows:

The iif owner of a (*,G) entry is the component owning the next-hop interface towards the nominal root of G, in the multicast RIB.

4.1. Interaction with the EGP

The fundamental requirements imposed by BGMP are that:

(1) For a given source-specific group and source, BGMP must be able to look up the next-hop towards the source in the Multicast RIB, and

(2) For a given non-source-specific group, BGMP will map the group address to a nominal "root" address, and must be able to look up the next-hop towards that address in the Multicast RIB.

BGMP determines the nominal "root" address as follows. If the multicast address is a Unicast-Prefix-based Multicast address, then the nominal root address is the embedded unicast prefix, padded with a suffix of 0 bits to form a full address.

For example, if the IPv6 group address is ff2e:0100:1234:5678:9abc:def0::123, then the unicast prefix is 1234:5678:9abc:def0/64, and the nominal root address would be 1234:5678:9abc:def0::. (This address is in fact the subnet router anycast address [IPv6AA].)

Support for any-source-multicast using any address other than a Unicast-prefix-based Multicast Address is outside the scope of this document.

4.2. Multicast Data Packet Processing

For BGMP rules to be applied, an incoming packet must first be "accepted":

- If the packet arrived on an interface owned by an M-IGP, the M-IGP component determines whether the packet should be accepted or dropped according to its rules. If the packet is accepted, the packet is forwarded (or not forwarded) out any other interfaces owned by the same component, as specified by the M-IGP.

- If the packet was received over a point-to-point interface owned by BGMP, the packet is accepted.

- If the packet arrived on a multiaccess network interface owned by BGMP, the packet is accepted if it is receiving data on a source-specific branch, if it is the designated forwarder for the longest matching route for S, or for the longest matching route for the nominal root of G.

If the packet is accepted, then the router checks the tree state table for a matching (S,G) entry. If one is found, but the packet was not received from the next hop target towards S (if the entry's SPT bit is True), or was not received from the next hop target towards G (if the entry's SPT bit is False) then the packet is dropped and no further actions are taken. If no (S,G) entry was found, the router then checks for a matching (*,G) entry.

If neither is found, then the packet is forwarded towards the next-hop peer for the nominal root of G, according to the Multicast RIB. If a matching entry was found, the packet is forwarded to all other targets in the target list.

Forwarding to a target which is an M-IGP component means that the packet is forwarded out any interfaces owned by that component according to that component's multicast forwarding rules.

4.3. BGMP processing of Join and Prune messages and notifications

4.3.1. Receiving Joins

When the BGMP component receives a (*,G) or (S,G) Join alert from another component, or a BGMP (S,G) or (*,G) Join message from an external peer, it searches the tree state table for a matching entry. If an entry is found, and that peer is already listed in the target list, then no further actions are taken.

Otherwise, if no (*,G) or (S,G) entry was found, one is created. In the case of a (*,G), the target list is initialized to contain the next-hop peer towards the nominal root of G, if it is an external peer. If the peer is internal, the target list is initialized to contain the M-IGP component owning the next-hop interface. If there is no next-hop peer (because the nominal root of G is inside the domain), then the target list is initialized to contain the next-hop component. If an (S,G) entry exists for the same G for which the (*,G) Join is being processed, and the next-hop peers toward S and the nominal root of G are different, the BGMP router must first send a (S,G) Prune message toward the source and clear the SPT bit on the (S,G) entry, before activating the (*,G) entry.

When creating (S,G) state, if the source is internal to the BGMP speaker's domain, a "Poison-Reverse" bit (PR-bit) is set. This bit indicates that the router may receive packets matching (S,G) anyway due to the BGMP speaker being a member of a domain on the path between S and the root domain. (Depending on the M-IGP protocol, it may in fact receive such packets anyway only if it is the best exit for the nominal root of G.)

The target from which the Join was received is then added to the target list. The router then looks up S or the nominal root of G in the Multicast RIB to find the next-hop EGP peer. If the target list, not including the next-hop target towards G for a (*,G) entry, becomes non-null as a result, the next-hop EGP peer must be notified as follows:

a) If the next-hop peer towards the nominal root of G (for a (*,G) entry) is an external peer, a BGMP (*,G) Join message is unicast to the external peer. If the next-hop peer towards S (for an (S,G) entry) is an external peer, and the router does NOT have any active (*,G) state for that group address G, a BGMP (S,G) Join message is unicast to the external peer. A BGMP (S,G) Join

message is never sent to an external peer by a router that also contains active (*,G) state for the same group. If the next-hop peer towards S (for an (S,G entry) is an external peer and the router DOES have active (*,G) state for that group G, the SPT bit is always set to False.

b) If the next-hop peer is an internal peer, a (*,G) or (S,G) Join alert is sent to the M-IGP component owning the next-hop interface.

c) If there is no next-hop peer, a (*,G) or (S,G) Join alert is sent to the M-IGP component owning the next-hop interface.

Finally, if an (S,G) Join is received from an internal peer, the peer should be stored with the M-IGP component target. If (S,G) state exists with the PR-bit set, and the next-hop towards the nominal root for G is through the M-IGP component, an (S,G) Poison-Reverse message is immediately sent to the internal peer.

If an (S,G) Join is received from an external peer, and (S,G) state exists with the PR-bit set, and the local BGMP speaker is the best exit for the nominal root of G, and the next-hop towards the nominal root for G is through the interface towards the external peer, an (S,G) Poison-Reverse message is immediately sent to the external peer.

4.3.2. Receiving Prune Notifications

When the BGMP component receives a (*,G) or (S,G) Prune alert from another component, or a BGMP (*,G) or (S,G) Prune message from an external peer, it searches the tree state table for a matching entry. If no (S,G) entry was found for an (S,G) Prune, but (*,G) state exists, an (S,G) entry is created, with the target list copied from the (*,G) entry. If no matching entry exists, or if the component or peer is not listed in the target list, no further actions are taken.

Otherwise, the component or peer is removed from the target list. If the target list becomes null as a result, the next-hop peer towards the nominal root of G (for a (*,G) entry), or towards S (for an (S,G) entry if and only if the BGMP router does NOT have any corresponding (*,G) entry), must be notified as follows.

a) If the peer is an external peer, a BGMP (*,G) or (S,G) Prune message is unicast to it.

b) If the next-hop peer is an internal peer, a (*,G) or (S,G) Prune alert is sent to the M-IGP component owning the next-hop interface.

c) If there is no next-hop peer, a (*,G) or (S,G) Prune alert is sent to the M-IGP component owning the next-hop interface.

4.3.3. Receiving Route Change Notifications

When a border router receives a route for a new prefix in the multicast RIB, or a existing route for a prefix is withdrawn, a route change notification for that prefix must be sent to the BGMP component. In addition, when the next hop peer (according to the multicast RIB) changes, a route change notification for that prefix must be sent to the BGMP component.

In addition, in IPv4 (only), an internal route for each class-D prefix associated with the domain (if any) MUST be injected into the multicast RIB in the EGP by the domain's border routers.

When a route for a new group prefix is learned, or an existing route for a group prefix is withdrawn, or the next-hop peer for a group prefix changes, a BGMP router updates all affected (*,G) target lists. The router sends a (*,G) Join to the new next-hop target, and a (*,G) Prune to the old next-hop target, as appropriate. In addition, if any (S,G) state exists with the PR-bit set:

- If the BGMP speaker has just become the best exit for the nominal root of G, an (S,G) Poison Reverse message with the PR-bit set is sent as noted below.

- If the BGMP speaker was the best exit for the nominal root of G and is no longer, an (S,G) Poison Reverse message with the PR-bit clear is sent as noted below.

The (S,G) Poison-Reverse messages are sent to all external peers on the next-hop interface towards the nominal root of G from which (S,G) Joins have been received.

When an existing route for a source prefix is withdrawn, or the next-hop peer for a source prefix changes, a BGMP router updates all affected (S,G) target lists. The router sends a (S,G) Join to the new next-hop target, and a (S,G) Prune to the old next-hop target, as appropriate.

4.3.4. Receiving (S,G) Poison-Reverse messages

When a BGMP speaker receives an (S,G) Poison-Reverse message from a peer, it sets the PR-bit on the (S,G) state to match the PR-bit in the message, and looks up the next-hop towards the nominal root of G. If the next-hop target is an M-IGP component, it forwards the (S,G) Poison Reverse message to all internal peers of that component from

which it has received (S,G) Joins. If the next-hop target is an external peer on a given interface, it forwards the (S,G) Poison Reverse message to all external peers on that interface.

When a BGMP speaker receives an (S,G) Poison-Reverse message from an external peer, with the PR-bit set, and the speaker has received no (S,G) Joins from any other peers (e.g., only from the M-IGP, or has (S,G) state due to encapsulation as described in 5.4.1), it knows that its own (S,G) Join is unnecessary, and should send an (S,G) Prune.

When a BGMP speaker receives an (S,G) Poison-Reverse message from an internal peer, with the PR-bit set, and the speaker is the best exit for the nominal root of G, and has (S,G) prune state, an (S,G) Join message is sent to cancel the prune state and the state is deleted.

4.4. Interaction with M-IGP components

When an M-IGP component on a border router first learns that there are internally-reached members for a group G (whose scope is larger than that domain), a (*,G) Join alert is sent to the BGMP component. Similarly, when an M-IGP component on a border router learns that there are no longer internally-reached members for a group G (whose scope is larger than a single domain), a (*,G) Prune alert is sent to the BGMP component.

At any time, any M-IGP domain MAY decide to join a source-specific branch for some external source S and group G. When the M-IGP component in the border router that is the next-hop router for a particular source S learns that a receiver wishes to receive data from S on a source-specific path, an (S,G) Join alert is sent to the BGMP component. When it is learned that such receivers no longer exist, an (S,G) Prune alert is sent to the BGMP component. Recall that the BGMP component will generate external source-specific Joins only where the source-specific branch does not coincide with the shared tree distribution tree for that group.

Finally, we will require that the border router that is the next-hop internal peer for a particular address S or the nominal root of G be able to forward data for a matching tree state table entry to all members within the domain. This requirement has implications on specific M-IGPs as follows.

4.4.1. Interaction with DVMRP and PIM-DM

DVMRP and PIM-DM are both "broadcast and prune" protocols in which every data packet must pass an RPF check against the packet's source address, or be dropped. If the border router receiving packets from an external source is the only BR to inject the route for the source into the domain, then there are no problems. For example, this will always be true for stub domains with a single border router (see Figure 1). Otherwise, the border router receiving packets externally is responsible for encapsulating the data to any other border routers that must inject the data into the domain for RPF checks to succeed.

When an intended border router injector for a source receives encapsulated packets from another border router in its domain, it should create source-specific (S,G) BGMP state. Note that the border router may be configured to do this on a data-rate triggered basis so that the state is not created for very low data-rate/intermittent sources. If source-specific state is created, then its incoming interface points to the virtual encapsulation interface from the border router that forwarded the packet, and it has an SPT flag that is initialized to be False.

When the (S,G) BGMP state is created, the BGMP component will in turn send a BGMP (S,G) Join message to the next-hop external peer towards S if there is no (*,G) state for that same group, G. The (S,G) BGMP state will have the SPT bit set to False if (*,G) BGMP state is present.

When the first data packet from S arrives from the external peer and matches on the BGMP (S,G) state, and IF there is no (*,G) state, the router sets the SPT flag to True, resets the incoming interface to point to the external peer, and sends a BGMP (S,G) Prune message to the border router that was encapsulating the packets (e.g., in Figure 1, BR11 sends the (Src_A,G) Prune to BR12). When the border router with (*,G) state receives the prune for (S,G), it then deletes that border router from its list of targets.

If the decapsulator receives a (S,G) Poison Reverse message with the PR-bit set, it will forward it to the encapsulator (which may again forward it up the shared tree according to normal BGMP rules), and both will delete their BGMP (S,G) state.

PIM-DM and DVMRP present an additional problem, i.e., no protocol mechanism exists for joining and pruning entire groups; only joins and prunes for individual sources are available. As a result, BGMP does not currently support such protocols being used in a transit domain.

4.4.2. Interaction with PIM-SM

Protocols such as PIM-SM build unidirectional shared and source-specific trees. As with DVMRP and PIM-DM, every data packet must pass an RPF check against some group-specific or source-specific address.

The fewest encapsulations/decapsulations will be done when the intra-domain tree is rooted at the next-hop internal peer (which becomes the RP) towards the nominal root of G, since in general that router will receive the most packets from external sources. To achieve this, each BGMP border router to a PIM-SM domain should send Candidate-RP-Advertisements within the domain for those groups for which it is the shared-domain tree ingress router. When the border router that is the RP for a group G receives an external data packet, it forwards the packet according to the M-IGP (i.e., PIM-SM) shared-tree outgoing interface list.

Other border routers will receive data packets from external sources that are farther down the bidirectional tree of domains. When a border router that is not the RP receives an external packet for which it does not have a source-specific entry, the border router treats it like a local source by creating (S,G) state with a Register flag set, based on normal PIM-SM rules; the Border router then encapsulates the data packets in PIM-SM Registers and unicasts them to the RP for the group. As explained above, the RP for the inter-domain group will be one of the other border routers of the domain.

If a source's data rate is high enough, DRs within the PIM-SM domain may switch to the shortest path tree. If the shortest path to an external source is via the group's ingress router for the shared tree, the new (S,G) state in the BGMP border router will not cause BGMP (S,G) Joins because that border router will already have (*,G) state. If however, the shortest path to an external source is via some other border router, that border router will create (S,G) BGMP state in response to the M-IGP (S,G) Join alert. In this case, because there is no local (*,G) state to suppress it, the border router will send a BGMP (S,G) Join to the next-hop external peer towards S, in order to pull the data down directly. (See BR11 in Figure 1). As in normal PIM-SM operation, those PIM-SM routers that have (*,G) and (S,G) state pointing to different incoming interfaces will prune that source off the shared tree. Therefore, all internal interfaces may be eventually pruned off the internal shared tree.

After the border router sends a BGMP (S,G) Join, if its (S,G) state has the PR-bit clear, a (S,G) Poison-Reverse message (with the PR-bit clear) is sent to the ingress router for G. The ingress router then creates (S,G) if it does not already exist, and removes the next hop towards the nominal root of G from the target list.

If the border router later receives an (S,G) Poison-Reverse message with the PR-bit set, the Poison-Reverse message is forwarded to the ingress router for G. The best-exit router then creates (S,G) state if it does not already exist, and puts the next hop towards the nominal root of G in the target list if not already present.

4.4.3. Interaction with CBT

CBT builds bidirectional shared trees but must address two points of compatibility with BGMP. First, CBT can not accommodate more than one border router injecting a packet. Therefore, if a CBT domain does have multiple external connections, the M-IGP components of the border routers are responsible for insuring that only one of them will inject data from any given source.

Second, CBT cannot process source-specific Joins or Prunes. Two options thus exist for each CBT domain:

Option A:

The CBT component interprets a (S,G) Join alert as if it were an (*,G) Join alert, as described in [INTEROP]. That is, if it is not already on the core-tree for G, then it sends a CBT (*,G) JOIN-REQUEST message towards the core for G. Similarly, when the CBT component receives an (S,G) Prune alert, and the child interface list for a group is NULL, then it sends a (*,G) QUIT_NOTIFICATION towards the core for G. This option has the disadvantage of pulling all data for the group G down to the CBT domain when no members exist.

Option B:

The CBT domain does not propagate any routes to their external peers for the Multicast RIB unless it is known that no other path exists to that prefix (e.g., routes for prefixes internal to the domain or in a singly-homed customer's domain may be propagated). This insures that source-specific joins are never received unless the source's data already passes through the domain on the shared tree, in which case the (S,G) Join need not be propagated anyway. BGMP border routers will only send source-specific Joins or Prunes to an external peer if that external peer advertises source-prefixes in the EGP. If a BGMP-CBT border router does receive an (S,G) Join or Prune, that border router should ignore the message.

To minimize en/de-capsulations, CBTv2 BR's may follow the same scheme as described under PIM-SM above, in which Candidate-Core advertisements are sent for those groups for which it is the shared-tree ingress router.

4.4.4. Interaction with MOSPF

As with CBTv2, MOSPF cannot process source-specific Joins or Prunes, and the same two options are available. Therefore, an MOSPF domain may either:

Option A:
send a Group-Membership-LSA for all of G in response to a (S,G) Join alert, and "prematurely age" it out (when no other downstream members exist) in response to an (S,G) Prune alert, OR

Option B:
not propagate any routes to their external peers for the Multicast RIB unless it is known that no other path exists to that prefix (e.g., routes for prefixes internal to the domain or in a singly-homed customer's domain may be propagated)

4.5. Operation over Multi-access Networks

Multiaccess links require special handling to prevent duplicates. The following mechanism enables BGMP to operate over multiaccess links which do not run an M-IGP. This avoids broadcast-and-prune behavior and does not require (S,G) state.

To elect a designated forwarder per prefix, BGMP uses a FWDR_PREF message to exchange "forwarder preference" values for each prefix. The peer with the highest forwarder preference becomes the designated forwarder, with ties broken by lowest BGMP Identifier. The designated forwarder is the router responsible for forwarding packets up the tree, and is the peer to which joins will be sent.

When BGMP first learns that a route exists in the multicast RIB whose next-hop interface is NOT the multiaccess link, the BGMP router sends a BGMP FWDR_PREF message for the prefix, to all BGMP peers on the LAN. The FWDR_PREF message contains a "forwarder preference value" for the local router, and the same value MUST be sent to all peers on the LAN. Likewise, when the prefix is no longer reachable, a FWDR_PREF of 0 is sent to all peers on the LAN.

Whenever a BGMP router calculates the next-hop peer towards a particular address, and that peer is reached over a BGMP-owned multiaccess LAN, the designated forwarder is used instead.

When a BGMP router receives a FWDR_PREF message from a peer, it looks up the matching route in its multicast RIB, and calculates the new designated forwarder. If the router has tree state entries whose parent target was the old forwarder, it sends Joins to the new forwarder and Prunes to the old forwarder.

When a BGMP router which is NOT the designated forwarder receives a packet on the multiaccess link, it is silently dropped.

Finally, this mechanism prevents duplicates where full peering exists on a "logical" link. Where full peering does not exist, steps must be taken (outside of BGMP) to present separate logical interfaces to BGMP, each of which is a link with full peering. This might entail, for example, using different link-layer address mappings, doing encapsulation, or changing the physical media.

4.6. Interaction between (S,G) state and G-routes

As discussed earlier, routers with (*,G) state will not propagate (S,G) joins. However, a special case occurs when (S,G) state coincides with the G-route (or route towards the nominal root of G). When this occurs, care must be taken so that the data will reach the root domain without causing duplicates or black holes. For this reason, (S,G) state on the path between the source and the root domain is annotated as being "poison-reversed". A PR-bit is kept for this purpose, which is updated by (UN)POISON_REVERSE messages.

The PR-bit indicates to BGMP nodes whether they need to forward packets up towards the root domain. For example, in a case where an (S,G) branch exists, a transit domain may get packets along the (S,G) branch, and needs to know whether to (also) forward them up towards the root domain. If the domain in question is on the path between S and the root domain, then the answer is yes (and the PR bit will be set on the S,G state). If the domain in question is not on the path between S and the root domain, then the answer is no (and the PR bit will be clear on the S,G state).

5. Message Formats

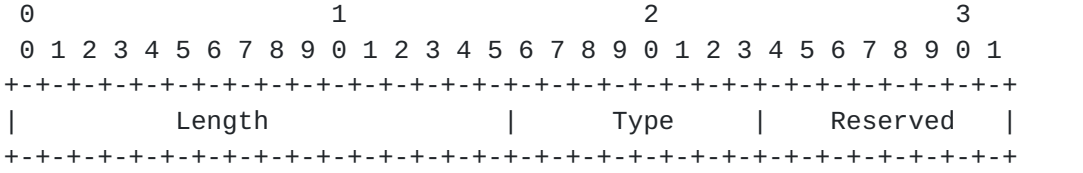
This section describes message formats used by BGMP.

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.

All fields labelled "Reserved" below must be transmitted as 0, and ignored upon receipt.

5.1. Message Header Format

Each message has a fixed-size (4-byte) 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:



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 start of the next message. The value of the Length field must always be at least 4 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.

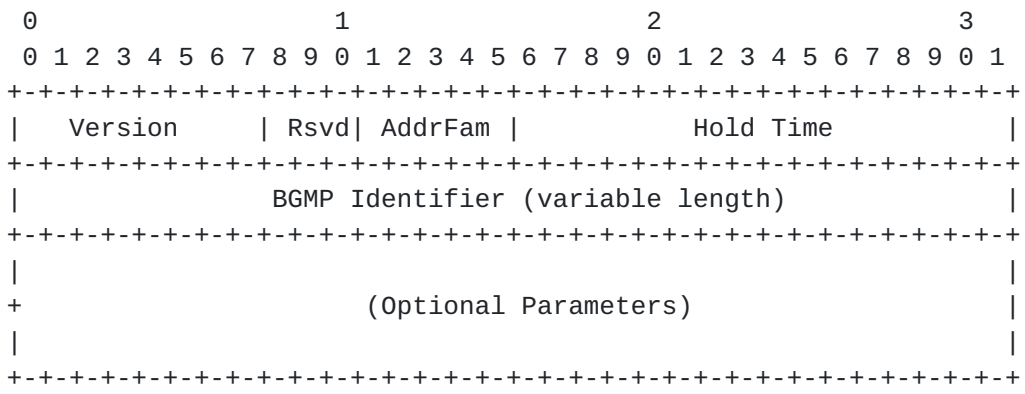
Type:
This 1-octet unsigned integer indicates the type code of the message. The following type codes are defined:

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

5.2. 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 BGMP header, the OPEN message contains the following fields:



Version:
   This 1-octet unsigned integer indicates the protocol version number of the message. The current BGMP version number is 1.

AddrFam:
   The IANA-assigned address family number of the BGMP Identifier. These include (among others):

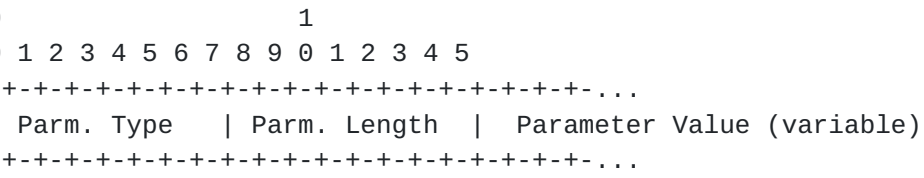
Number	Description
1	IP (IP version 4)
2	IPv6 (IP version 6)

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 BGMP 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.

BGMP Identifier:
   This 4-octet (for IPv4) or 16-octet (IPv6) unsigned integer indicates the BGMP Identifier of the sender. A given BGMP speaker sets the value of its BGMP Identifier to a globally-unique value assigned to that BGMP speaker (e.g., an IPv4 address). The value of the BGMP Identifier is determined on startup and is the same for every BGMP session opened.

Optional Parameters:

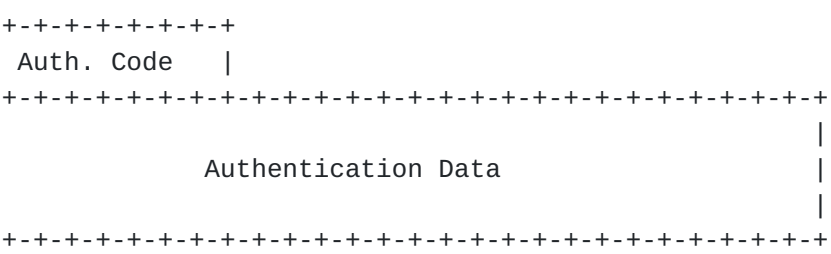
This field may contain a list of optional parameters, where each parameter is encoded as a <Parameter Length, Parameter Type, Parameter Value> triplet. The combined length of all optional parameters can be derived from the Length field in the message header.



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.

This document defines the following Optional Parameters:

a) Authentication Information (Parameter Type 1): This optional parameter may be used to authenticate a BGMP peer. The Parameter Value field contains a 1-octet Authentication Code followed by a variable length Authentication Data.



Authentication Code:

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

- the value of the Authentication Code which indicates use of the mechanism, and - the form and meaning of the Authentication Data.

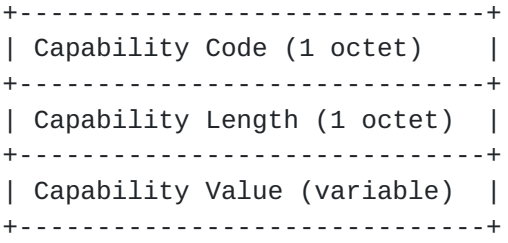
Note that a separate authentication mechanism may be used in establishing the transport level connection.

Authentication Data:

The form and meaning of this field is a variable-length field depend on the Authentication Code.

The minimum length of the OPEN message is 12 octets (including message header).

b) Capability Information (Parameter Type 2): This is an Optional Parameter that is used by a BGMP-speaker to convey to its peer the list of capabilities supported by the speaker. The parameter contains one or more triples <Capability Code, Capability Length, Capability Value>, where each triple is encoded as shown below:



Capability Code:

Capability Code is a one octet field that unambiguously identifies individual capabilities.

Capability Length:

Capability Length is a one octet field that contains the length of the Capability Value field in octets.

Capability Value:

Capability Value is a variable length field that is interpreted according to the value of the Capability Code field.

A particular capability, as identified by its Capability Code, may occur more than once within the Optional Parameter.

This document reserves Capability Codes 128-255 for vendor-specific applications.

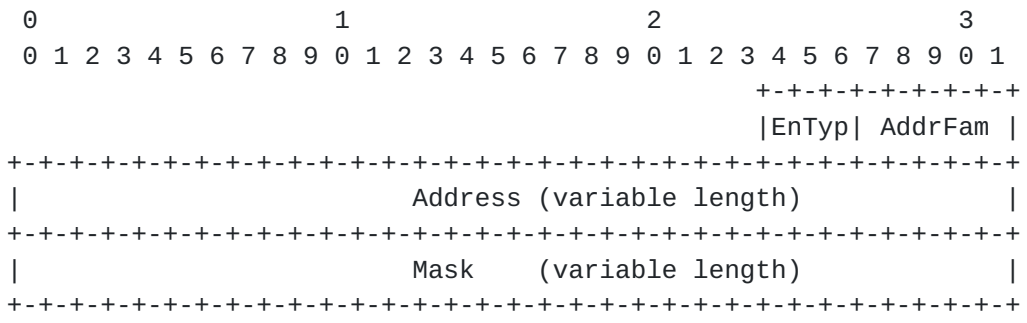
This document reserves value 0.

Capability Codes (other than those reserved for vendor specific use) are assigned only by the IETF consensus process and IESG approval.

5.3. UPDATE Message Format

UPDATE messages are used to transfer Join/Prune/FwdrPref information between BGMP peers. The UPDATE message always includes the fixed-size BGMP header, and one or more attributes as described below.

The message format below allows compact encoding of (*,G) Joins and Prunes, while allowing the flexibility needed to do other updates such as (S,G) Joins and Prunes towards sources as well as on the shared tree. In the discussion below, an Encoded-Address-Prefix is of the form:



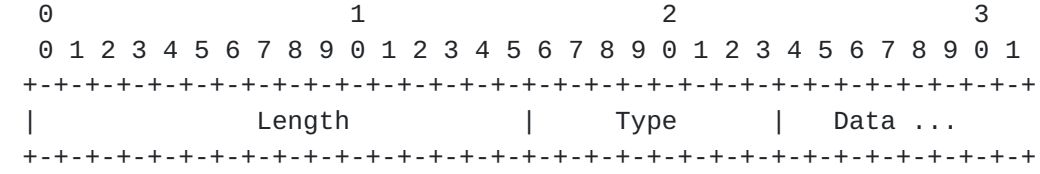
EnTyp:
- 0 - All 1's Mask. The Mask field is 0 bytes long.
- 1 - Mask length included. The Mask field is 4 bytes long, and contains the mask length, in bits.
- 2 - Full Mask included. The Mask field is the same length as the Address field, and contains the full bitmask.

AddrFam:
The IANA-assigned address family number of the encoded prefix.

Address:
The address associated with the given prefix to be encoded. The length is determined based on the Address Family.

Mask:
The mask associated with the given prefix. The format (or absence) of this field is determined by the EnTyp field.

Each attribute is of the form:



All attributes are 4-byte aligned.

Length:
The Length is the length of the entire attribute, including the length, type, and data fields. If other attributes are nested within the data field, the length includes the size of all such nested attributes.

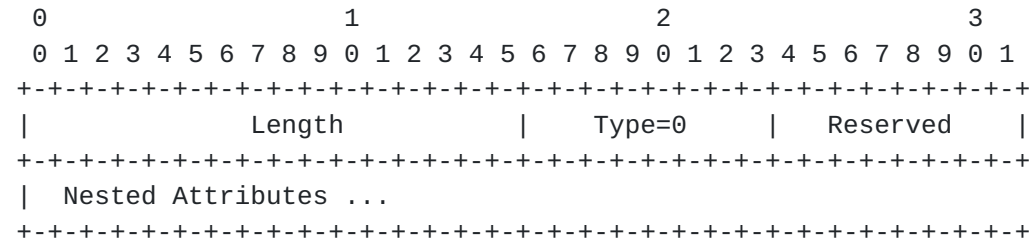
Type:

Types 128-255 are reserved for "optional" attributes. If a required attribute is unrecognized, a NOTIFICATION will be sent and the connection will be closed if the error is a fatal one. Unrecognized optional attributes are simply ignored.

0 - JOIN
1 - PRUNE
2 - GROUP
3 - SOURCE
4 - FWDR_PREF
5 - POISON_REVERSE

a) JOIN (Type Code 0)

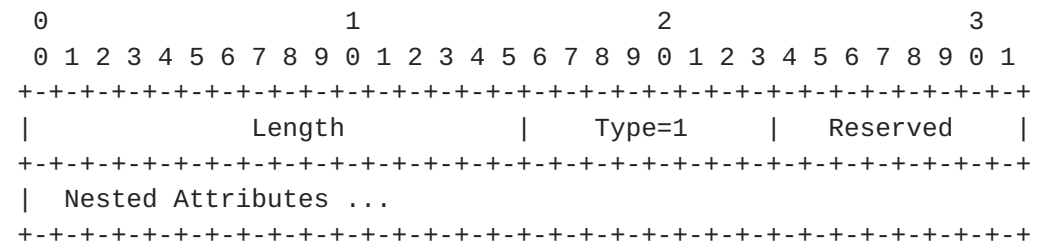
The JOIN attribute indicates that all GROUP or SOURCE options nested immediately within the JOIN option should be joined.



No JOIN, PRUNE, or FWDR_PREF attributes may be immediately nested within a JOIN attribute.

b) PRUNE (Type Code 1)

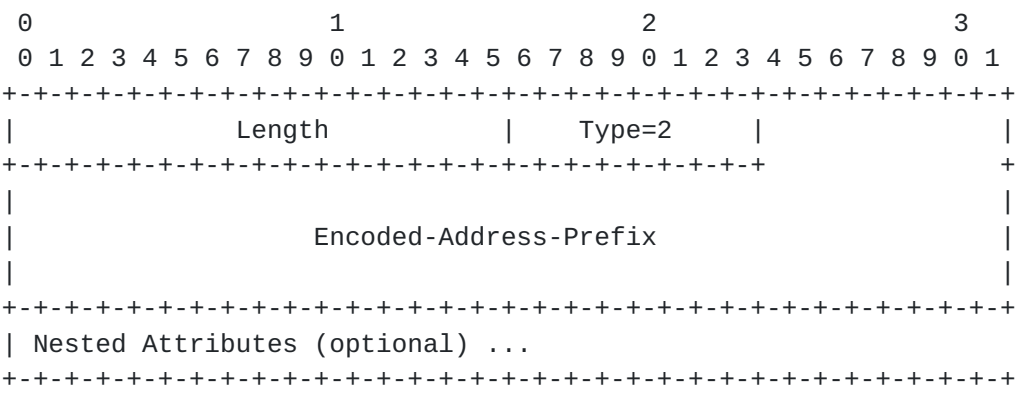
The PRUNE attribute indicates that all GROUP or SOURCE attributes nested immediately within the PRUNE attribute should be pruned.



No JOIN, PRUNE, or FWDR_PREF attributes may be immediately nested within a PRUNE attribute.

c) GROUP (Type Code 2)

The GROUP attribute identifies a given group-prefix. In addition, any attributes nested immediately within the GROUP attribute also apply to the given group-prefix.



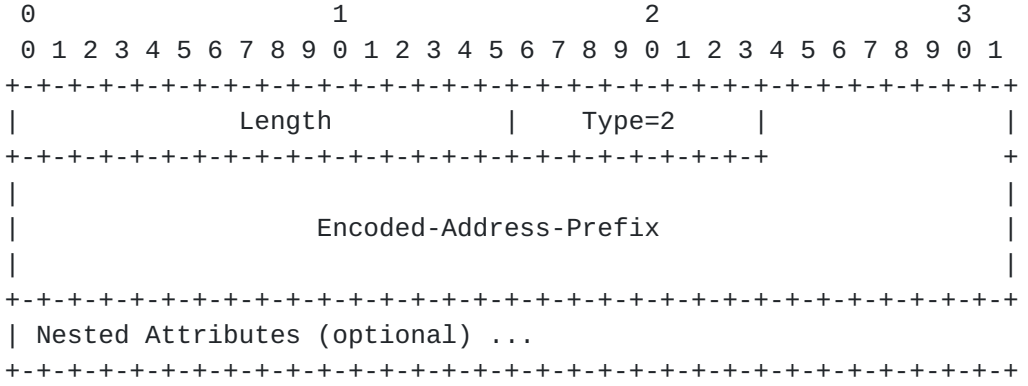
Encoded-Address-Prefix The multicast group prefix to be joined to pruned, in the format described above.

Nested Attributes No GROUP, SOURCE, or FWDR_PREF attributes may be immediately nested within a GROUP attribute.

d) SOURCE (Type Code 3):

The SOURCE attribute identifies a given source-prefix. In addition, any attributes nested immediately within the SOURCE attribute also apply to the given source-prefix.

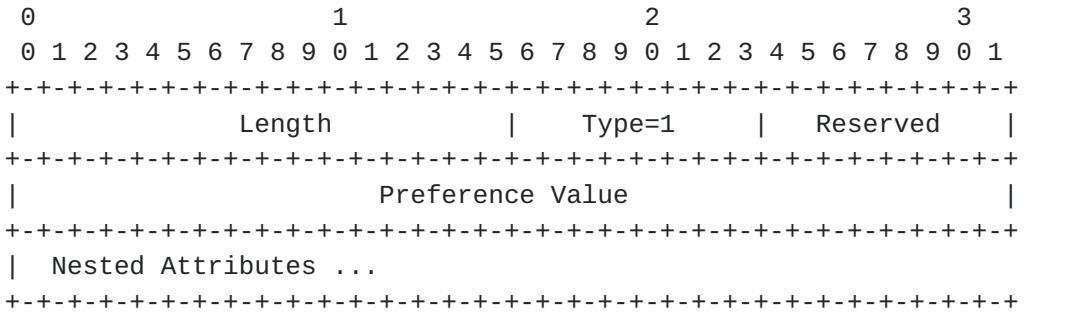
The SOURCE attribute has the following format:



Encoded-Address-Prefix	The Source-prefix in the format described above.
Nested Attributes	No GROUP, SOURCE, or FWDR_PREF attributes may be immediately nested within a SOURCE attribute.

e) FWDR_PREF (Type Code 4)

The FWDR_PREF attribute provides a forwarder preference value for all GROUP or SOURCE attributes nested immediately within the FWDR_PREF attribute. It is used by a BGMP speaker to inform other BGMP speakers of the originating speaker's degree of preference for a given group or source prefix. Usage of this attribute is described in 5.5.

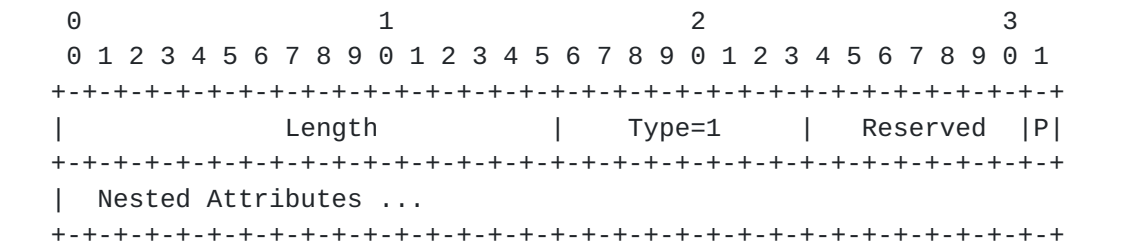


Preference Value	A 32-bit non-negative integer.
Nested Attributes	No JOIN, PRUNE, or FWDR_PREF attributes may be immediately nested within a FWDR_PREF attribute.

e) POISON_REVERSE (Type Code 5)

The POISON_REVERSE attribute provides a "poison-reverse" (PR-bit) value for all SOURCE attributes nested immediately within the POISON_REVERSE attribute. It is used by a BGMP speaker to inform

other BGMP speakers from which it has received (S,G) Joins that they are on the path of domains between the source and the root domain.



P	The PR-bit value.
Nested Attributes	No attributes in the document other than SOURCE may be immediately nested within a POISON_REVERSE attribute.

5.4. Encoding examples

Below are enumerated examples of how various updates are built using nested attributes, where A (B) denotes that attribute B is nested within attribute A.

(*,G-prefix) Join: JOIN (GROUP)
(*,G-prefix) Prune: PRUNE (GROUP)
(S,G) Join towards S : GROUP (JOIN (SOURCE))
(S,G) Join cancelling prune towards root of G: GROUP (JOIN (SOURCE))
(S,G) Prune towards S: GROUP (PRUNE (SOURCE))
(S,G) Prune towards root of G: GROUP (PRUNE (SOURCE))
Switch from (*,G) to (S,G): PRUNE (GROUP (JOIN (SOURCE)))
Switch from (S,G) to (*,G): JOIN (GROUP)
Initial (*,G) Join with S pruned: JOIN (GROUP (PRUNE (SOURCE)))
Forwarder preference announcement for G-prefix: FWDR_PREF (GROUP)
Forwarder preference announcement for S-prefix: FWDR_PREF (SOURCE)

5.5. KEEPALIVE Message Format

BGMP 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 Timer to expire. A reasonable maximum time between the last KEEPALIVE or UPDATE message sent, and the time at which a KEEPALIVE message is sent, 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.

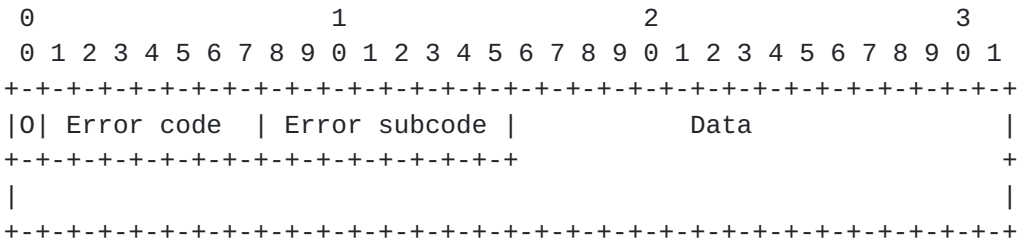
If the negotiated Hold Time interval is zero, then periodic KEEPALIVE messages MUST NOT be sent.

A KEEPALIVE message consists of only a message header, and has a length of 4 octets.

5.6. NOTIFICATION Message Format

A NOTIFICATION message is sent when an error condition is detected. The BGMP connection is closed immediately after sending it if the error is a fatal one.

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



O-bit:
Open-bit. If clear, the connection will be closed. If set, indicates the error is not fatal.

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 9.1
2	OPEN Message Error	Section 9.2
3	UPDATE Message Error	Section 9.3
4	Hold Timer Expired	Section 9.5
5	Finite State Machine Error	Section 9.6
6	Cease	Section 9.7

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. The notation (MC) below indicates the error is a fatal one and the O-bit must be clear. Non-fatal subcodes SHOULD be sent with the O-bit set.

Message Header Error subcodes:

2 - Bad Message Length (MC)
3 - Bad Message Type (MC)

OPEN Message Error subcodes:

1 - Unsupported Version (MC)
4 - Unsupported Optional Parameter
5 - Authentication Failure (MC)
6 - Unacceptable Hold Time (MC)
7 - Unsupported Capability (MC)

UPDATE Message Error subcodes:

1 - Malformed Attribute List (MC)
2 - Unrecognized Attribute Type
5 - Attribute Length Error (MC)
10 - Invalid Address
11 - Invalid Mask
13 - Unrecognized Address Family

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 7 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 = 6 + Data Length

The minimum length of the NOTIFICATION message is 6 octets (including message header).

6. BGMP Error Handling

This section describes actions to be taken when errors are detected while processing BGMP messages. BGMP Error Handling is similar to that of BGP [BGP].

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 BGMP connection is closed if the error is a fatal one. If no Error Subcode is specified, then a zero must be used.

The phrase "the BGMP connection is closed" means that the transport protocol connection has been closed and that all resources for that BGMP connection have been deallocated. The remote peer is removed from the target list of all tree state entries.

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.

If the Length field of the message header is less than 4 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 4, 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 BGMP peer bid (as indicated in the received OPEN message).

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 one of the Optional Parameters in the OPEN message is not recognized, then the Error Subcode is set to Unsupported Optional Parameters.

If the OPEN message carries Authentication Information (as an Optional Parameter), 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.

If the OPEN message indicates that the peer does not support a capability which the receiver requires, the receiver may send a NOTIFICATION message to the peer, and terminate peering. The Error Subcode in the message is set to Unsupported Capability. The Data field in the NOTIFICATION message lists the set of capabilities that cause the speaker to send the message. Each such capability is encoded the same way as it was encoded in the received OPEN message.

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.

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 the Encoded-Address-Prefix field in some attribute is syntactically incorrect, then the Error Subcode is set to Invalid Prefix Field.

If any other is encountered when processing attributes (such as invalid nestings), then the Error Subcode is set to Malformed Attribute List, and the problematic attribute is included in the data 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 must be sent and the BGMP connection closed.

6.6. Finite State Machine error handling

Any error detected by the BGMP 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 BGMP peer may choose at any given time to close its BGMP 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.

6.8. Connection collision detection

If a pair of BGMP speakers try simultaneously to establish a TCP connection to each other, then two parallel connections between this pair of speakers might well be formed. We refer to this situation as connection collision. Clearly, one of these connections must be closed.

Based on the value of the BGMP Identifier a convention is established for detecting which BGMP connection is to be preserved when a collision does occur. The convention is to compare the BGMP

Identifiers of the peers involved in the collision and to retain only the connection initiated by the BGMP speaker with the higher-valued BGMP Identifier.

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

1. The BGMP Identifier of the local system is compared to the BGMP Identifier of the remote system (as specified in the OPEN message).

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

3. Otherwise, the local system closes newly created BGMP 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).

Comparing BGMP Identifiers is done by treating them as (4-octet long) unsigned integers.

A connection collision with an existing BGMP connection that is in Established states causes unconditional closing of the newly created connection. Note that a connection collision cannot be detected with connections that are in Idle, or Connect, or Active states.

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

7. BGMP Version Negotiation

BGMP speakers may negotiate the version of the protocol by making multiple attempts to open a BGMP 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 BGMP 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 BGMP version negotiation, future versions of BGMP must retain the format of the OPEN and NOTIFICATION messages.

7.1. BGMP Capability Negotiation

When a BGMP speaker sends an OPEN message to its BGMP peer, the message may include an Optional Parameter, called Capabilities. The parameter lists the capabilities supported by the speaker.

A BGMP speaker may use a particular capability when peering with another speaker only if both speakers support that capability. A BGMP speaker determines the capabilities supported by its peer by examining the list of capabilities present in the Capabilities Optional Parameter carried by the OPEN message that the speaker receives from the peer.

8. BGMP Finite State machine

This section specifies BGMP operation in terms of a Finite State Machine (FSM). Following is a brief summary and overview of BGMP operations by state as determined by this FSM.

Initially BGMP is in the Idle state.

Idle state:

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

If a BGMP speaker detects an error, it shuts down the connection and changes its state to Idle. Getting out of the Idle state requires generation of the Start event. If such an event is generated automatically, then persistent BGMP errors may result in persistent flapping of the speaker. To avoid such a condition it is recommended that Start events should not be generated immediately for a peer that was previously transitioned to Idle due to an error. For a peer that was previously transitioned to Idle due to an error, the time between consecutive generation of

Start events, if such events are generated automatically, shall exponentially increase. The value of the initial timer shall be 60 seconds. The time shall be doubled for each consecutive retry.

Any other event received in the Idle state is ignored.

Connect state:

In this state BGMP 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 BGMP 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 the other BGMP peer, continues to listen for a connection that may be initiated by the remote BGMP peer, and stays in the Connect state.

The Start event is ignored in the Connect state.

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

Active state:

In this state BGMP is trying to acquire a peer by listening for an incoming 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. A Hold Timer value of 4 minutes is suggested.

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

If the local system detects that a remote peer is trying to establish BGMP 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 BGMP peer, and stays in the Active state.

The 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 BGMP resources associated with this connection and changes its state to Idle.

OpenSent state:

In this state BGMP waits for an OPEN message from its peer. When an OPEN message is received, all fields are checked for correctness. If the BGMP message header checking or OPEN message checking detects an error (see Section 6.2), or a connection collision (see Section 6.8) the local system sends a NOTIFICATION message and changes its state to Idle.

If there are no errors in the OPEN message, BGMP sends a KEEPALIVE message and sets a KeepAlive timer. The Hold Timer, which was originally set to a large value (see above), is replaced with the negotiated Hold Time value (see section 4.2). If the negotiated Hold Time value is zero, then the Hold Time timer and KeepAlive timers are not started. If the configured remote Autonomous System value for this peering is the same as the local Autonomous System number, then the connection is an "internal" connection; otherwise, it is "external". Finally, the state is changed to OpenConfirm.

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

If the Hold Timer 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.

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

OpenConfirm state:

In this state BGMP 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.

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

Established state:

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

If the local system receives an UPDATE or KEEPALIVE message, it restarts its Hold Timer, if the negotiated Hold Time value is non-zero.

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 Hold 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, unless the negotiated Hold Time value is zero.

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.

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

9. Security Considerations

If a BGMP speaker accepts unauthorized or altered BGMP messages, denial of service due to excess bandwidth consumption or lack of multicast connectivity can result. Authentication of BGMP messages can protect against this behavior.

A BGMP implementation MUST implement Keyed MD5 [RFC2385] to secure control messages, and MUST be capable of interoperating with peers that do not support it. However, if one side of the connection is configured with Keyed MD5 and the other side is not, the connection SHOULD NOT be established.

This provides a weak security mechanism, as it is still possible for denial of service to occur as a result of messages relayed through a trusted peer. However, this model is the same as the currently practiced security mechanism for BGP. It is anticipated that future work will provide different stronger mechanisms for dealing with these issues in routing protocols.

10. Acknowledgements

In addition to the editor, the following individuals have contributed to the design of BGMP: Cengiz Alaettinoglu, Tony Ballardie, Steve Casner, Steve Deering, Deborah Estrin, Dino Farinacci, Bill Fenner, Mark Handley, Ahmed Helmy, Van Jacobson, Dave Meyer, and Satish Kumar.

This document is the product of the IETF BGMP Working Group with Dave Thaler as editor.

Rusty Eddy, Isidor Kouvelas, and Pavlin Radoslavov also provided valuable feedback on this document.

11. References

11.1. Normative References

[INTEROP] Thaler, D., "Interoperability Rules for Multicast Routing Protocols", RFC 2715, October 1999.

[RFC2385] Heffernan, A., "Protection of BGP sessions via the TCP MD5 Signature Option", RFC 2385, August 1998.

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

[V6PREFIX] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.

11.2. Informative References

[BGP] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995.

[MBGP] Bates, T., Rekhter, Y., Chandra, R., and D. Katz, "Multiprotocol Extensions for BGP-4", RFC 2858, June 2000.

[CBT] Ballardie, A., "Core Based Trees (CBT version 2) Multicast Routing -- Protocol Specification", RFC 2189, September 1997.

[DVMRP] Pusateri, T., "Distance Vector Multicast Routing Protocol", Work in Progress, October 2003.

[IPv6AA] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[MOSPF] Moy, J., "Multicast Extensions to OSPF", RFC 1584, March 1994.

[PIMDM] Adams, A., Nicholas, J. and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", Work in Progress, September 2003.

[PIMSM] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P., and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2362, June 1998.

[REFLECT] Bates, T. and R. Chandra, "BGP Route Reflection: An alternative to full mesh IBGP", RFC 1966, June 1996.

[V4PREFIX] Thaler, D., "Unicast-Prefix-based IPv4 Multicast Addresses", Work in Progress, August 2004.

Authors' Address

Dave Thaler
Microsoft
One Microsoft Way
Redmond, WA 98052

EMail: dthaler@microsoft.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.