Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 20 September 2026

D. Farinacci
lispers.net
M. McBride
Futurewei

19 March 2026

Group Address Allocation Protocol (GAAP)
draft-ietf-pim-gaap-11

Abstract

This document describes a design for a lightweight decentralized multicast group address allocation protocol (named GAAP and pronounced "gap" as in "mind the gap"). The protocol requires no configuration setup and no centralized services. The protocol runs among group participants which need a unique group address to send and receive multicast packets. Tailored for IPv4 and IPv6 networks, this design offers a simple, lightweight option rather than extending an existing protocol.

Status of This Memo

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

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

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

This Internet-Draft will expire on 20 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The Group Address Allocation Protocol (GAAP) is a decentralized multicast protocol used by participating applications which send and receive packets to/from a multicast group. The protocol is relatively lightweight, runs with minimized messaging and state so that it can run within a library a multicast application compiles into its executable binary.

GAAP is a possible solution to the issues described in problem statement draft [I-D.ietf-pim-zeroconf-mcast-addr-alloc-ps].

Other approaches to multicast group allocation have been proposed in the past, they include SAP [RFC2974], SDP [RFC4566], mDNS [RFC6762], MADCAP [RFC2730], MASC [RFC2909], and IPv6 Allocation Guidelines [RFC3307]. However, they require global scope adding latency, configuration, use of a single subnet, and are not decentralized.

This document will describe the protocol operation, protocol message formats, the API definition, and how multicast applications use the API.

2. Definition of Terms

Group Name: is an ASCII string used by applications so they can rendezvous on the same group address. The application is started using this group name parameter. Applications can use multiple group names if they have requirements to use multiple group addresses.

Group Address: is a non-link-local IPv4 multicast group address [RFC1112] or an IPv6 multicast group address [RFC4291].

GAAP Group Address: is an IANA assigned non-link-local group address the GAAP protocol sends messages to. This address must not come from the GAAP multicast address block allocated by IANA. For IPv4, the range is TBD1/10. For IPv6, the range is TBD2/32. See Section 8 for IANA considerations.

Hash Function: is a cryptographic hash function which takes the group name as input and produces a hash value as output. The GAAP protocol uses SHA-256 [RFC6234].

Acceptable Group Hash List: There are 4 hashed values regarded as

"acceptable" for a group name. They are calculated using the SHA-256 hash function on 1 of 4 character strings: "<group-name>", "<group-name>+1", "<group-name>+2", or "<group-name>+3". No GAAP node should run the hash on any other strings for this group name.

Hashed Value: is the output of a SHA-256 [RFC6234] hash function where the low order 32-bits are used to produce a unique network layer multicast group address. Achieving a unique 32-bits allows layer-2 switches to not have MAC multicast address collisions when mapped from multiple network layer multicast group addresses.

Collided Group Address: a network layer group address where the low-order 32-bits of one group address is the same as the low-order 32-bits of another group address. It is desirable that the low-order 32-bits of a mapped IPv6 group address to a MAC group address not be the same so layer-2 switches do not leak packets to non-group members. This is also true for IPv4 group addresses where the low-order 23-bits must be unique.

Claim Message: a GAAP protocol message that allocates a unique group address and claims it among other GAAP nodes on the network.

3. Overview of Protocol Operation

This section will describe the high-level functionality of the GAAP protocol. Each application runs the GAAP protocol by using the API defined in Section 5.

* An application is started with a group name.

* The group name is used to create a random allocated group address.

* A timestamp is taken when the group address is created.

* A Claim message, see Section 4, is sent with group name, group address, and timestamp to determine if the group address has been claimed by any other GAAP nodes.

* If no Claim message is sent in response, the application can start using the group address.

* If a Claim message is returned, a collision has occurred and the GAAP node must allocate another group address and send a Claim message for the new group address.

* Claim messages are sent periodically. They are sent by a single node using a delay-timer suppression mechanism similar to IGMP [RFC1112]. See Section 6 for details.

* GAAP nodes are not required to cache information from Claim messages.

* GAAP is designed to be decentralized and stateless. The nodes that participate in the GAAP protocol are responsible for allocating and claiming group addresses. No other entities are needed.

4. GAAP Message Format

At this time, there is a single message called the Claim message with type value 1. Type value of 0 is reserved. The Claim message is sent in a UDP checksummed packet where the source port is ephemeral and chosen by the sender and the destination port is a well-known port allocated by IANA. GAAP can work behind NAT and firewall devices as long as the GAAP destination port is permitted through filters.

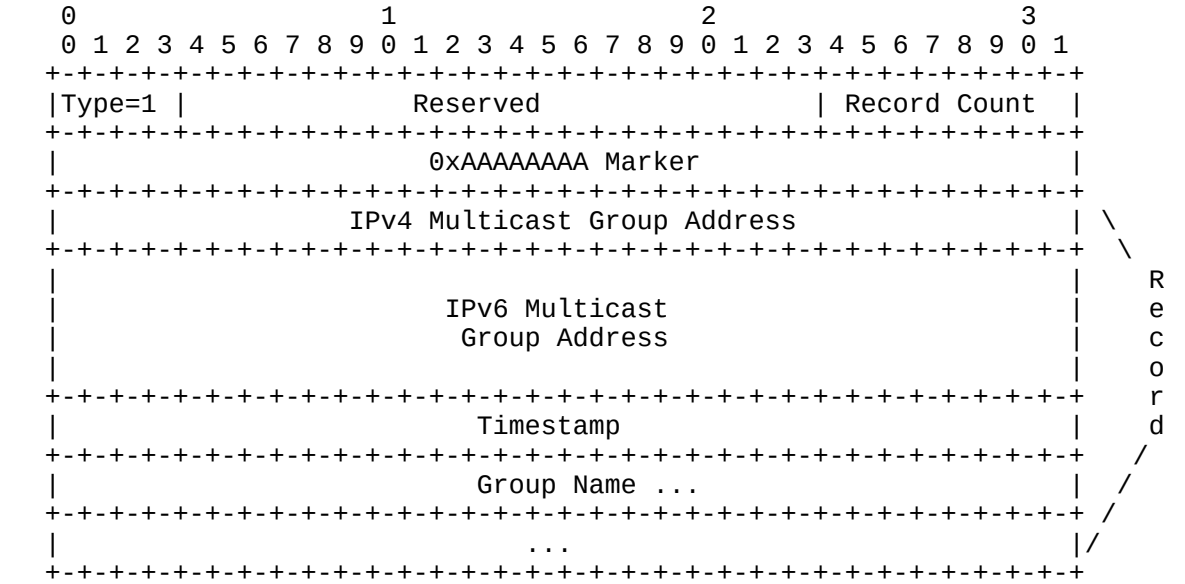

Figure 1: GAAP Claim Message

Packet field descriptions:

Type=1: Claim Message

Reserved: Set to 0 and ignored on receipt.

Record Count: The number of records in this Claim message.

Marker: The fixed bit pattern of 0xAAAAAAAA is required to be set by the sender. The receiver verifies the marker to be 0xAAAAAAAA. If it is not, the packet is dropped. The Marker field is used to indicate to a receiver that the packet may be encrypted. See Section 7 for details on encrypting GAAP messages.

Record field descriptions:

IPv4 Multicast Group Address: a 32-bit multicast address in network byte order [RFC1112]. If all bits are set to 0, there is no IPv4 address being allocated and claimed.

IPv6 Multicast Group Address: a 128-bit multicast address in network byte order [RFC4291]. If all bits are set to 0, there is no IPv6 address being allocated and claimed.

Timestamp: A 32-bit standard epoch UTC timestamp in seconds according to [RFC8536].

Group Name: A variable length group name the multicast application uses. It is in ASCII format [RFC0020]. The string is terminated with a null character. Since the Group Name is variable length, subsequent records may not occur on a long- or short-word boundary.

5. GAAP API

The GAAP API has the following API calls a multicast application will use. A multicast application imports the library before using it in its code logic. This section documents a python library.

5.1. gaap.init()

gaap.init() is used to initialize the GAAP API with a application callback function. The callback function is called when a group address has changed (due to collision) for a group name the application allocated.

```
            import gaap

            status = gaap.init(app_callback_func)
            if (status == False):
                print("error")
                exit(1)
            #endif

            def app_callback_func(group_name, group_address)
                print("Group name {} changed to group address {}". \
                    format((group_name, group_address))
            #enddef
```

5.2. gaap.allocate()

gaap.allocate() is used when the application needs a group address to send or receive on.

```
        import gaap

        group_name = "my-audio-group"

        group_address = gaap.allocate(group_name)
        if (group_address == None):
            print("error")
            exit(1)
        #endif

        print("Name {} allocated address {}".format(group_name, group_address))
```

5.3. gaap.release()

gaap.release() is used when an application is finished using a group address.

```
            import gaap

            group_address = gaap.allocate("my-audio-group")

            status = gaap.release(group_address)
            if (status == False):
                print("error")
                exit(1)
            #endif

            print("Released address {}".format(group_address))
```

5.4. gaap.close()

gaap.close() is used when an application is finished using the GAAP protocol.

```
import gaap

#
# Initialize the GAAP API with no callback function. Return if errored.
#
if (gaap.init() == False):
    print("error")
    exit(1)
#endif

#
# Do multicast work by allocating, sending, and receiving group addresses.
#
...

#
# Application shutting down. No longer need to run GAAP on local node.
#
gaap.close()
```

6. Detail Protocol Operation

6.1. Allocating Group Addresses

When an application needs a group address it provides the GAAP API with a group name, the group name is used as input to a SHA-256 hash function [RFC6234]. Initially, when no group address collision is detected the group name is passed as a string to the hash function and the low-order 32-bits are used for a group address. The following pseudo-code illustrates the functionality:

```
hash = sha256(group_name)
low_bits = hash & 0xffffffff
if (v4):
    group_address = 0xe0000000 | (low_bits & 0x007fffff)
#endif
if (v6):
    group_address = 0xff0e...0000 | low_bits
#endif
return(group_address)
```

When the hash function is used to resolve a collision, the following pseudo-code will illustrate how 3 more attempts are used to find a unique group address:

```
for append in ["+1", "+2", "+3"]:
    hash = sha256(group_name + append)
    group_address = make_group_from_hash(hash)
    collision = send_claim(group_address)
    if (collision == False): return
#endfor
print("Collision limit reached")
```

When a group address collision is detected by 2 GAAP nodes, the node with the earliest timestamp for the group address creation wins the collision and keeps using the address. The node with a later timestamp has the responsibility to allocate a new group address to prevent the collision.

6.2. Claiming Group Addresses

When a group address is allocated by a GAAP node it will build and send a Claim message. Included in the Claim message is the group name, group address, and timestamp. If the group address collides with other GAAP nodes already using the address, one of the nodes will send a Claim message to notify the colliding node that it needs to allocate a new group address.

A collision is defined to be the same group address allocated to 2 different group names. So if a GAAP node is claiming a group address for its group name and a Claim is received with the same group name with the same group address, it is not a collision. It is simply a peer group participant claiming the group address you both agree to be using.

Each GAAP node will periodically send Claim messages for all group names for the applications running on the node. It will do this in a multi-record Claim message. The periodic Claim message is sent by setting a rough 1 minute timer. The timer value is set to 1 minute plus a jitter value. The jitter value is a random number in a 10% range of 1 minute (60 to 66 seconds). When the timer expires, a Claim message is sent. Receivers of a Claim message who have their timer running, reset the timer and thereby suppresses sending their own Claim message. This allows only a single GAAP node that is using the group address to keep claiming the group is still in use.

A new GAAP node may come up after a group address collision has been resolved. It may send a Claim message for the first group hash from the Acceptable Group Hash List. When this happens previous nodes in

the group will trigger sending a Claim for one of the other addresses in the Acceptable Group Hash List. The new GAAP node must switch over to using the triggered Claimed group address. The new GAAP node must yield to existing nodes since their timestamps for the group address allocation happened earlier than the new node's Claim.

6.3. Partition Repair

There will be network outage situations where all GAAP nodes may not receive Claim messages. During a partition, duplicate group addresses may be allocated and used by nodes on each side of the partition. During this condition, multicast nodes can operate normally and there is no conflict until the partition heals. When the partition heals, duplicate group addresses will be detected and fixed. The group address with the earliest timestamp is used to determine who keeps the collided group address. All others will have to rehash a new group address and have the applications start using the new address (meaning senders will source using the new group address and receivers will leave the collided group and join the new group).

6.4. Releasing Group Addresses

When applications are no longer sending to a group address or not joined to a group address, they can inform the GAAP API to release the group. When this happens, the GAAP protocol stops claiming the group address in periodic messages and will not respond to a Claim for this address for a different group name. It is important for receiver applications to leave the group before releasing the group address.

7. Security Considerations

It is strongly suggested that the GAAP protocol run over an encrypted multicast channel. All GAAP implementations should support the same encryption mechanism and use the same key management procedure to ensure interoperability. This could be difficult in embedded devices with different configurations. The message Marker is used to indicate if the packet is sent in plaintext or ciphertext. If the Marker is not set to 0xAAAAAAAA and the receiver does not have a shared-key configured, the message MUST be dropped.

An open-source GAAP implementation exists where ChaCha20 [RFC7539] is used to encrypt GAAP messages. The implementation's key management procedure is a simple shared key that is configured with the application.

At this time there is no dynamic rekeying procedure and is left for future work. Therefore, all nodes must be manually rekeyed when a node is removed from the encrypted channel.

The following attack threats may exist with possible mitigation techniques:

* Even when an encrypted channel is used, a bad actor could be claiming a group address not derived from one of the group name inputs used for the Acceptable Group Hash List (see Definition of Terms section). Cooperating nodes should ignore such messages and not try to send Claim messages to correct the bad actor node.

* A bad actor could send an invalid timestamp giving it tie-breaking priority when a group address collision occurs. If the group address has been prior claimed by another node with a timestamp earlier than the invalid timestamp, cooperating nodes should put the bad actor node on a bad-actor list and ignore future messages from it. If the group name has not been claimed yet, the timestamp will be accepted only if earlier than the current time for the receiving node.

* A bad actor could send messages too often and is not adhering to the random delay or periodic timer procedures in this document. When this occurs, cooperating nodes should start ignoring messages from the bad actor node and not reset or cancel timers, or send triggered Claim messages

8. IANA Considerations

IANA will create the following registry in a new registry group called "Group Address Allocation Protocol (GAAP)":

Registry Name: Group Address Allocation Protocol (GAAP)
Registration Procedure: IETF Review

8.1. GAAP UDP Port Numbers

IANA will create one UDP port number for the GAAP protocol:

Service Name	Port Number	Transport Protocol	Description	Reference
gaap	TBD	udp	GAAP Control Packets	draft-ietf-pim-gaap

Table 1

8.2. GAAP Protocol Multicast Addresses

IANA will create one multicast address from the IPv4 Internetwork Control Block 224.0.1.x and one multicast address from the IPv6 Variable Scope Multicast Addresses Block FF0X::TBD for the operation of the GAAP protocol. The registry description field should indicate "GAAP".

8.3. GAAP Multicast Group Allocation Ranges

IANA will create two multicast address ranges for the GAAP protocol to allocate application-use addresses from. For IPv4, a /10 block in a new registry range is requested. A /10 block represents a large portion of the IPv4 multicast space and discussion between IETF and IANA is needed to determine the appropriate block size. This is a significant allocation that impacts the overall IPv4 multicast address space. For IPv6, a /32 block in a new registry range is being requested. This allocation should come from the Dynamic Multicast Group IDs registry defined in [I-D.ietf-pim-updt-ipv6-dyn-mcast-addr-grp-id].

Registry Name: GAAP IPv4 Allocation Range
Registration Procedure: IETF Review

Registry Name: GAAP IPv6 Allocation Range
Registration Procedure: IETF Review

For IPv6 multicast addresses, the GAAP application allocation range should be in the new "Dynamic Multicast Group IDs" registry requested by [I-D.ietf-pim-updt-ipv6-dyn-mcast-addr-grp-id]. This new registry requests the division of the 32-bit group ID range 0xA0000000 through 0xAFFFFFFF. The GAAP allocation range should come out of this 32-bit range.

9. References

9.1. Normative References

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC8536] Olson, A., Eggert, P., and K. Murchison, "The Time Zone Information Format (TZif)", RFC 8536, DOI 10.17487/RFC8536, February 2019, <https://www.rfc-editor.org/info/rfc8536>.

9.2. Informative References

[I-D.ietf-pim-updt-ipv6-dyn-mcast-addr-grp-id] Karstens, N., Farinacci, D., and M. McBride, "Updates to Dynamic IPv6 Multicast Address Group IDs", Work in Progress, Internet-Draft, draft-ietf-pim-updt-ipv6-dyn-mcast-addr-grp-id-13, 10 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-pim-updt-ipv6-dyn-mcast-addr-grp-id-13>.

[I-D.ietf-pim-zeroconf-mcast-addr-alloc-ps] Karstens, N., Farinacci, D., and M. McBride, "Zeroconf Multicast Address Allocation Problem Statement and Requirements", Work in Progress, Internet-Draft, draft-ietf-pim-zeroconf-mcast-addr-alloc-ps-13, 17 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-pim-zeroconf-mcast-addr-alloc-ps-13>.

[RFC2730] Hanna, S., Patel, B., and M. Shah, "Multicast Address Dynamic Client Allocation Protocol (MADCAP)", RFC 2730, DOI 10.17487/RFC2730, December 1999, <https://www.rfc-editor.org/info/rfc2730>.

[RFC2909] Radoslavov, P., Estrin, D., Govindan, R., Handley, M., Kumar, S., and D. Thaler, "The Multicast Address-Set Claim (MASC) Protocol", RFC 2909, DOI 10.17487/RFC2909, September 2000, <https://www.rfc-editor.org/info/rfc2909>.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <https://www.rfc-editor.org/info/rfc2974>.

[RFC3307] Haberman, B., "Allocation Guidelines for IPv6 Multicast Addresses", RFC 3307, DOI 10.17487/RFC3307, August 2002, <https://www.rfc-editor.org/info/rfc3307>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015, <https://www.rfc-editor.org/info/rfc7539>.

Appendix A. Acknowledgments

The authors would like to thank the following people for their motivation to start this draft. They include Chris Hopps, Acee Lindem, David Lamparter, Jeff Tantsura, Nate Karstens, and Lenny Giuliano.

Appendix B. Document Change Log

B.1. Changes to draft-ietf-pim-gaap-11

* Submitted March 2026.

* Change IANA request for an IPv4 multicast block from /8 to /10.

* Changes made by Dino.

B.2. Changes to draft-ietf-pim-gaap-10

* Submitted February 2026.

* Incorporated suggestion from Toerless Eckert to add paragraph discussing SAP and SDP approaches to multicast group allocation in the Introduction section.

* Added references to RFC2974 (SAP) and RFC4566 (SDP).

* Changes made by Dino.

B.3. Changes to draft-ietf-pim-gaap-09

* Submitted February 2026.

* Be more clear about variable length group names and how subsequent records may not be aligned.

* Changes made by Dino.

B.4. Changes to draft-ietf-pim-gaap-08

* Submitted February 2026.

* Incorporated WG review comments from Stig Venaas on GAAP Group Address definition, Timestamp field documentation, encryption interoperability requirements, and IANA allocation discussion.

* Changes made by Dino.

B.5. Changes to draft-ietf-pim-gaap-07

* Submitted January 2026.

* Change draft name in IANA considerations section to point to the IETF draft name and not the individual contribution draft name.

* Changes made by Dino.

B.6. Changes to draft-ietf-pim-gaap-06

* Submitted September 2025.

* Dino fixes Kasten reference.

B.7. Changes to draft-ietf-pim-gaap-05

* Submitted September 2025.

* Mike adds one-liner to abstract.

B.8. Changes to draft-ietf-pim-gaap-04

* Submitted August 2025.
* Fix some typos in the GAAP API section.

B.9. Changes to draft-ietf-pim-gaap-03

* Submitted February 2025.
* Fix some typos in the GAAP API section.
* Update references and docuemnt timer.

B.10. Changes to draft-ietf-pim-gaap-02

* Submitted September 2024.
* Update references and docuemnt timer.

B.11. Changes to draft-ietf-pim-gaap-01

* Submitted April 2024.
* Update references and docuemnt timer.

B.12. Changes to draft-ietf-pim-gaap-00

* Submitted October 2023.
* Made draft-farinacci-pim-gaap-06 into WG document per PIM WG consensus.

B.13. Changes to draft-farinacci-pim-gaap-06

* Submitted September 2023.
* Fix Nate last name misspelling.
* Add reference to [I-D.ietf-pim-zeroconf-mcast-addr-alloc-ps] to intro section.
* In the IANA Considerations, add IPv6 allocation for GAAP in the 0xA0000000-0xAFFFFFFF range, as suggested by Nate.

B.14. Changes to draft-farinacci-pim-gaap-05

* Submitted August 2023.

* Update IANA Considerations section to have IPv6 GAAP application allocations come from the registry that [I-D.ietf-pim-updt-ipv6-dyn-mcast-addr-grp-id] is creating.

B.15. Changes to draft-farinacci-pim-gaap-04

* Submitted April 2023.

* Added specific text recommended by IANA in the IANA Considerations section.

B.16. Changes to draft-farinacci-pim-gaap-03

* Submitted April 2023.

* Changes to reflect comments from PIM and MBONED WG meetings.

* Put IANA Considerations requests in standard request format.

B.17. Changes to draft-farinacci-pim-gaap-02

* Submitted March 2023.

* Fix typos and grammer.

B.18. Changes to draft-farinacci-pim-gaap-01

* Submitted February 2023.

* Updated spec to reflect implementation.

* Add Marker in message format.

* Add definition for the Acceptable Group Hash List.

* Discuss security threats and possible mitigation methods.

B.19. Changes to draft-farinacci-pim-gaap-00

* Initial posting November 2022.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
United States of America
Email: farinacci@gmail.com

Mike McBride
Futurewei
Santa Clara, CA
United States of America
Email: mmcbride7@gmail.com