

Multicast Ping Protocol

Abstract

The Multicast Ping Protocol specified in this document allows for checking whether an endpoint can receive multicast -- both Source-Specific Multicast (SSM) and Any-Source Multicast (ASM). It can also be used to obtain additional multicast-related information, such as multicast tree setup time. This protocol is based on an implementation of tools called "ssmping" and "asmping".

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc6450>.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	2
2. Architecture	3
3. Protocol Specification	6
3.1. Option Format	7
3.2. Defined Options	7
3.3. Packet Format	13
3.4. Message Types and Options	13
3.5. Rate Limiting	15
3.5.1. Message Rate Variables	16
4. Client Behaviour	16
5. Server Behaviour	18
6. Recommendations for Implementers	19
7. IANA Considerations	20
8. Security Considerations	21
9. Acknowledgments	22
10. References	23
10.1. Normative References	23
10.2. Informative References	23

1. Introduction

The Multicast Ping Protocol specified in this document allows for checking multicast connectivity. In addition to checking reception of multicast (SSM or ASM), the protocol can provide related information, such as multicast tree setup time, the number of hops the packets have traveled, and packet delay and loss. This functionality resembles, in part, the ICMP Echo Request/Reply mechanism [RFC0792], but uses UDP [RFC0768] and requires that both a client and a server implement this protocol. Intermediate routers are not required to support this protocol. They forward protocol messages and data traffic as usual.

This protocol is based on the current implementation of the ssmping and asmping tools [IMPL], which are widely used by the Internet community to conduct multicast connectivity tests.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Architecture

Before describing the protocol in detail, we provide a brief overview of how the protocol may be used and what information it may provide.

The protocol is used between clients and servers to check multicast connectivity. Servers are multicast sources, and clients are multicast receivers. A server may be configured with a set of ranges of multicast addresses that can be used for testing, or it may use some implementation defaults. Depending on the server configuration or the implementation, it may control which clients (which unicast addresses) are allowed to use different group ranges, and also whether clients can select a group address, or if the group address is selected by the server. Whether several clients are allowed to simultaneously use the same multicast address also depends on configuration and/or implementation.

In addition to the above state, a server normally has runtime soft state. The server must generally perform rate limiting to restrict the number of client requests it handles. This rate limiting is per-client IP address. This state need usually only be maintained for a few seconds, depending on the limit used. If the server provides unique multicast addresses to clients, it must also have soft state for tracking which multicast addresses are used by which client IP address. This state should expire if the server has not received requests within a few minutes. The exact timeout should ideally be configurable to cope with different environments. If a client is expected to perform multicast ping checks continuously for a long period of time, and to cope with requests not reaching the client for several minutes, then this timeout needs to be extended. In order to verify the client IP address, the server should perform a return routability check by giving the client a non-predictable session ID. This would then also be part of the server soft state for that client.

Before it can perform a multicast ping test, a client must know the unicast address of a server. In addition, it may be configured with a multicast address or range to use. In that case, the client will tell the server which group or range it wishes to use. If not, the server is left to decide the group. Normally, a client sends Default-Client-Request-Rate requests per second. It may, however, be configured to use another rate. See the definition of Default-Client-Request-Rate in Section 3.5.1. Note that the value can be less than 1.

At runtime, a client generates a client ID that is unique for the ping test. This ID is included in all messages sent by the client. Further, if not supplied with a specific group address, the client will receive from the server a group address that is used for the ping requests. It may also receive a Session ID from the server. The client ID, group address, and Session ID (if received) will then be fixed for all ping requests in this session. When a client receives replies from a server, it will verify the client ID in the reply, and ignore it if not matching what it used in the requests. For each reply, it may print or record information like round trip time, number of hops, etc. The client may, once a ping session is ended, calculate and print or record statistics based on the entire ping session.

The typical protocol usage is as follows:

A server runs continuously to serve requests from clients. A client has somehow learned the unicast address of the server and tests the multicast reception from the server.

The client application will then send a unicast message to the server, asking for a group to use. Optionally, a user may request a specific group or scope, in which case the client will ask for a group matching the user's request. The server will respond with a group to use, or an error if no group is available.

Next, for ASM, the client joins an ASM group G, while for SSM it joins a channel (S,G), where G is the multicast group address specified by the server, and S is the unicast address used to reach the server.

After joining the group/channel, the client unicasts multicast ping requests to the server. The requests are sent using UDP with the destination port set to the standardised multicast ping port (9903). The requests are sent periodically to the server. The rate is by default Default-Client-Request-Rate (Section 3.5.1) requests per second, but the client may be configured to use another rate. These requests contain a sequence number and, typically, a timestamp. The requests are echoed by the server, which may add a few options.

For each request, the server sends two replies. One reply is unicast to the source IP address and source UDP port of the requesting client. The other reply is multicast to the requested multicast group G and the source UDP port of the requesting client.

Both replies are sent from the same port on which the request was received. The server should specify the TTL (IPv4 time-to-live or IPv6 hop-count) used for both the unicast and multicast messages (TTL of at least 64 is recommended) by including a TTL option. This allows the client to compute the number of hops. The client should leave the group/channel when it has finished its measurements.

By use of this protocol, a client (or a user of the client) can obtain information about several multicast delivery characteristics. First, by receiving unicast replies, the client can verify that the server is receiving the unicast requests, and is operational and responding. Hence, provided that the client receives unicast replies, a failure to receive multicast indicates either a multicast problem or a multicast administrative restriction. If it does receive multicast, it knows not only that it can receive multicast traffic but that it may also estimate the amount of time it took to establish the multicast tree (at least if it is in the range of seconds), whether there are packet drops, and the length and variation of round trip times (RTTs).

For unicast, the RTT is the time from when the unicast request is sent to the time when the reply is received. The measured multicast RTT also references the client's unicast request. By specifying the TTL of the replies when they are originated, the client can also determine the number of router hops it is from the source. Since similar information is obtained in the unicast replies, the host may compare its multicast and unicast results and is able to check for differences, such as the number of hops, and RTT.

The number of multicast hops and changes in the number of hops over time may reveal details about the multicast tree and multicast tree changes. Provided that the server sends the unicast and multicast replies nearly simultaneously, the client may also be able to measure the difference in one-way delay for unicast and multicast on the path from server to client.

Servers may optionally specify a timestamp. This may be useful, since the unicast and multicast replies cannot be sent simultaneously (the delay is dependent on the host's operating system and load).

3. Protocol Specification

There are four different message types:

- o Echo Request and Echo Reply messages, which are used for the actual measurements.
- o An Init message, which **SHOULD** be used to initialise a ping session and negotiate which group to use.
- o A Server Response message, which is mainly used in response to the Init message.

The messages **MUST** always be in network byte order. UDP checksums **MUST** always be used.

The messages share a common format: one octet specifying the message type, followed by a number of options in TLV (Type, Length, and Value) format. This makes the protocol easily extendible.

Message types in the range 0-253 are reserved and available for allocation in an IANA registry. Message types 254 and 255 are freely available for experimental use. See Section 7.

The Init message generally contains some prefix options asking the server for a group from one of the specified prefixes. The server responds with a Server Response message that contains the group address to use, or possibly prefix options describing what multicast groups the server may be able to provide.

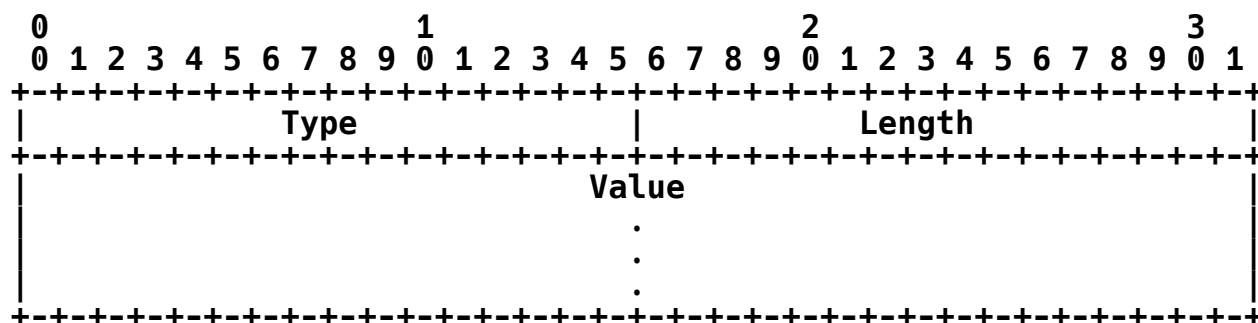
For an Echo Request, the client includes a number of options, and a server **MAY** simply echo the contents (only changing the message type) without inspecting the options if it does not support any options. This might be true for a simple Multicast Ping Protocol server, but it severely limits what information a client can obtain and hence makes the protocol less useful. However, the server **SHOULD** add a TTL option (allowing the client to determine the number of router hops a reply has traversed), and there are other options that a server implementation **MAY** support; e.g., the client may ask for certain information or a specific behaviour from the server. The Echo Reply messages (one unicast and one multicast) **MUST** first contain the exact options from the request (in the same order), and then, immediately following, any options appended by the server. A server **MUST NOT** process unknown options, but they **MUST** still be included in the Echo Reply. A client **MUST** ignore any unknown options.

The size of the protocol messages is generally smaller than the Path MTU, and fragmentation is not a concern. There may, however, be cases where the Path MTU is really small, or where a client sends large requests in order to verify that it can receive fragmented multicast datagrams. This document does not specify whether Path MTU Discovery should be performed, etc. A possible extension could be an option where a client requests Path MTU Discovery and receives the current Path MTU from the server.

This document defines a number of different options. Some options do not require processing by servers and are simply returned unmodified in the reply. There are, however, other client options that the server may care about, as well as server options that may be requested by a client. Unless otherwise specified, an option **MUST NOT** be used multiple times in the same message.

3.1. Option Format

All options are TLVs formatted as below.



Type (2 octets) specifies the option.

Length (2 octets) specifies the length of the value field. Depending on the option type, it can be from 0 to 65535.

Value must always be of the specified length. See the respective option definitions for possible values. If the length is 0, the value field is not included.

3.2. Defined Options

This document defines the following options: Version (0), Client ID (1), Sequence Number (2), Client Timestamp (3), Multicast Group (4), Option Request (5), Server Information (6), TTL (9), Multicast Prefix (10), Session ID (11), and Server Timestamp (12). Option values 7 and 8 are deprecated and must not be allocated by any future document. The options are defined below.

Option types in the range 0-65531 are reserved and available for allocation in an IANA registry. Option types in the range 65532-65535 are not registered and are freely available for experimental use. See Section 7.

Version, type 0

Length MUST be 1. This option MUST always be included in all messages, and for the current specified protocol this value MUST be set to 2 (in decimal). Note that there are implementations of older revisions of this protocol that only partly follow this specification. They can be regarded as version 1 and do not use this option. If a server receives a message with a version other than 2 (or missing), the server SHOULD (unless it supports the particular version) send a Server Response message back with version set to 2. This tells the client that the server expects version 2 messages. Client ID and Sequence Number options MUST be echoed if present, so that a client can be certain it is a response to one of its messages, and to exactly which message. The server SHOULD NOT include any other options. A client receiving a response with a version other than 2 MUST stop sending requests to the server (unless it supports the particular version).

Client ID, type 1

Length MUST be non-zero. A client SHOULD always include this option in all messages (both Init and Echo Request). The client may use any value it likes to detect whether a reply is a reply to its Init/Echo Request or not. A server should treat this as opaque data, and MUST echo this option back in the reply if present (both Server Response and Echo Reply). The value might be a pseudo-random byte string that is likely to be unique, possibly combined with the client IP address. Predictability is not a big concern here. This is used by the client to ensure that server messages are in response to its requests. In some cases, a client may receive multicast responses to queries from other clients. It is left to the client implementer how to use this option.

Sequence Number, type 2

Length MUST be 4. A client MUST always include this in Echo Request messages and MUST NOT include it in Init messages. A server replying to an Echo Request message MUST copy it into the Echo Reply (or Server Response message on error). The sequence number is a 32-bit integer. Values typically start at 1 and increase by one for each Echo Request in a sequence.

Client Timestamp, type 3

Length MUST be 8. A client SHOULD include this in Echo Request messages and MUST NOT include it in Init messages. A server replying to an Echo Request message MUST copy it into the Echo Reply. The timestamp specifies the time when the Echo Request message is sent. The first 4 bytes specify the number of seconds since the Epoch (0000 UTC Jan 1, 1970). The next 4 bytes specify the number of microseconds since the second specified in the first 4 bytes. This option would typically be used by a client to compute round trip times.

Note that while this protocol uses the above 32-bit format, it would have been better to use another format, such as the one defined in NTPv4 [RFC5905]. This should be considered for future extensions of the protocol.

Multicast Group, type 4

Length MUST be greater than 2. It MAY be used in Server Response messages to tell the client what group to use in subsequent Echo Request messages. It MUST be used in Echo Request messages to tell the server what group address to respond to (this group would typically be previously obtained in a Server Response message). It MUST be used in Echo Reply messages (copied from the Echo Request message). It MUST NOT be used in Init messages. The format of the option value is as below.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Address Family           | Multicast group address... |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      ....

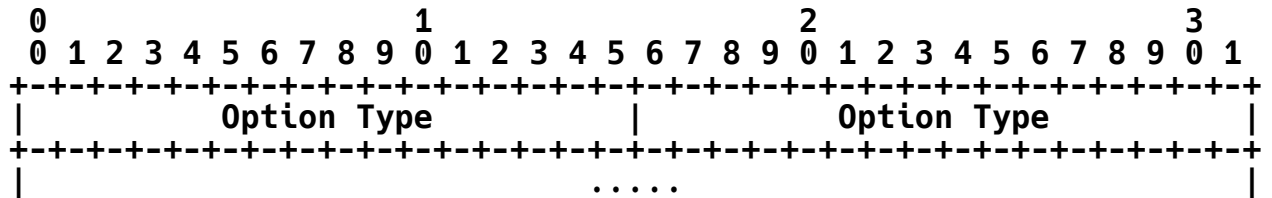
```

The address family is a value 0-65535 as assigned by IANA for Internet address families [ADDRFAMILY]. This is followed by the group address. Length of the option value will be 6 for IPv4, and 18 for IPv6.

Option Request, type 5

Length MUST be greater than 1. This option MAY be used in client messages (Init and Echo Request messages). A server MUST NOT send this option, except that if it is present in an Echo Request message, the server MUST echo it in replies (Echo Reply message) to the Echo Request. This option contains a list of option types for options that the client is requesting

from the server. Support for this option is OPTIONAL for both clients and servers. The length of this option will be a non-zero even number, since it contains one or more option types that are two octets each. The format of the option value is as below.



This option might be used by the client to ask the server to include options like Server Timestamp or Server Information. A client MAY request Server Information in Init messages; it MUST NOT request it in other messages. A client MAY request a Server Timestamp in Echo Request messages; it MUST NOT request it in other messages. Subject to enforcing the above restrictions, a server supporting this option SHOULD include the requested options in responses (Echo Reply messages) to the Echo Request containing the Option Request option. The server may, according to implementation or local configuration, not necessarily include all the requested options, or possibly none. Any options included are appended to the echoed options, similar to other options included by the server.

Server Information, type 6

Length MUST be non-zero. It MAY be used in Server Response messages and MUST NOT be used in other messages. Support for this option is OPTIONAL. A server supporting this option SHOULD add it in Server Response messages if and only if requested by the client. The value is a UTF-8 [RFC3629] string that might contain vendor and version information for the server implementation. It may also contain information on which options the server supports. An interactive client MAY support this option, and SHOULD then allow a user to request this string and display it. Although support for this is OPTIONAL, we say that a server SHOULD return it if requested, since this may be helpful to a user running the client. It is, however, purely informational; it is not needed for the protocol to function.

Deprecated, type 7

This option code value was used by implementations of version 1 of this protocol, and is not used in this version. Servers **MUST** treat it as an unknown option (not process it if received, but if received in an Echo Request message, it **MUST** be echoed in the Echo Reply message).

Deprecated, type 8

This option code value was used by implementations of version 1 of this protocol, and is not used in this version. Servers **MUST** treat it as an unknown option (not process it if received, but if received in an Echo Request message, it **MUST** be echoed in the Echo Reply message).

TTL, type 9

Length **MUST** be 1. This option contains a single octet specifying the TTL of an Echo Reply message. Every time a server sends a unicast or multicast Echo Reply message, it **SHOULD** include this option specifying the TTL. This is used by clients to determine the number of hops the messages have traversed. It **MUST NOT** be used in other messages. A server **SHOULD** specify this option if it knows what the TTL of the Echo Reply will be. In general, the server can specify a specific TTL to the host stack. Note that the TTL is not necessarily the same for unicast and multicast. Also note that this option **SHOULD** be included even when not requested by the client. The protocol will work even if this option is not included, but it limits what information a client can obtain.

If the server did not include this TTL option, there is no reliable way for the client to know the initial TTL of the Echo Reply, and therefore the client **SHOULD NOT** attempt to calculate the number of hops the message has traversed.

Multicast Prefix, type 10

Length **MUST** be greater than 2. It **MAY** be used in Init messages to request a group within the prefix(es), and it **MAY** be used in Server Response messages to tell the client from what prefix(es) it may try to obtain a group. It **MUST NOT** be used in Echo Request/Reply messages. Note that this option **MAY** be included multiple times to specify multiple prefixes.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Address Family           | Prefix Length | Partial address |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                                     ....

```

The address family is a value 0-65535 as assigned by IANA for Internet address families [ADDRFAMILY]. This is followed by a prefix length (4-32 for IPv4, 8-128 for IPv6, or 0 for the special "wildcard" use discussed below), and finally a group address. For any family, prefix length 0 means that any multicast address from that family is acceptable. This is what we call "wildcard". The group address need only contain enough octets to cover the prefix length bits (i.e., the group address would have to be 3 octets long if the prefix length is 17-24, and there need be no group address for the wildcard with prefix length 0). Any bits past the prefix length **MUST** be ignored. For IPv4, the option value length will be 4-7, while for IPv6, it will be 4-19, and for the wildcard, it will be 3.

Session ID, type 11

Length **MUST** be 4 or larger. A server **SHOULD** include this in Server Response messages. If a client receives this option in a message, the client **MUST** echo the Session ID option in subsequent Echo Request messages, with the exact same value. The Session ID may help the server in keeping track of clients and possibly manage per-client state. The value of a new Session ID **SHOULD** be a pseudo-random byte string that is hard to predict; see [RFC4086]. The string **MUST** be at least 4 bytes long. The Session ID can be used to mitigate spoofing of the source address of Echo Request messages. We say that this option **SHOULD** be used, because it is important for security reasons. There may, however, be environments where this is not required. See Section 8, "Security Considerations", for details.

Server Timestamp, type 12

Length **MUST** be 8 bytes. A server supporting this option **SHOULD** include it in Echo Reply messages, if requested by the client. The timestamp specifies the time when the Echo Reply message is sent. The first 4 bytes specify the number of seconds since the Epoch (0000 UTC Jan 1, 1970). The next 4 bytes specify the number of microseconds since the second specified in the first 4 bytes. If this option is not included, the protocol will still work, but it makes it impossible for a client to compute one-way delay.

Server Response, type 83

This message is sent by a server, either as a response to an Init, or in response to an Echo Request. When responding to an Init, it may provide the client with a multicast group (if requested by the client), or it may provide other server information. In response to an Echo Request, the message tells the client to stop sending Echo Request messages. The Version option **MUST** always be included. Client ID and Sequence Number options are echoed if present in the client message. When providing a group to the client, it includes a Multicast Group option. It **SHOULD** include Server Information and Prefix options if requested. It **SHOULD** also include the Session ID option.

Echo Request, type 81

This message is sent by a client, asking the server to send unicast and multicast Echo Reply messages. It **MUST** include Version, Sequence Number, and Multicast Group options. If a Session ID was received in a Server Response message, then the Session ID **MUST** be included. It **SHOULD** include Client ID and Client Timestamp options. It **MAY** include an Option Request option.

Echo Reply, type 65

This message is sent by a server in response to an Echo Request message. This message is always sent in pairs, one as unicast and one as multicast. The contents of the messages are mostly the same. The server always echoes all of the options (but never the Session ID) from the Echo Request. Any options in the Echo Request that are unsupported by the server are also to be echoed. The two Echo Reply messages **SHOULD** both always contain a TTL option (not necessarily equal). When requested, both Echo Reply messages **SHOULD** also contain Server Timestamps (not necessarily equal).

The matrix below summarises what options can go in what messages.

Option \ Message Type	Init	Server Response	Echo Request	Echo Reply
Version (0)	MUST	MUST	MUST	ECHO
Client ID (1)	SHOULD	ECHO	SHOULD	ECHO
Sequence Number (2)	NOT	ECHO	MUST	ECHO
Client Timestamp (3)	NOT	NOT	SHOULD	ECHO
Multicast Group (4)	NOT	MAY	MUST	ECHO
Option Request (5)	MAY	NOT	MAY	ECHO
Server Information (6)	NOT	RQ	NOT	NOT
Deprecated (7)	NOT	NOT	NOT	ECHO
Deprecated (8)	NOT	NOT	NOT	ECHO
TTL (9)	NOT	NOT	NOT	SHOULD
Multicast Prefix (10)	MAY	MAY	NOT	NOT
Session ID (11)	NOT	SHOULD	ECHO	NOT
Server Timestamp (12)	NOT	NOT	NOT	RQ

"NOT" means that the option MUST NOT be included. "ECHO" for a server means that if the option is specified by the client, then the server MUST echo the option in the response, with the exact same option value. ECHO for a client only applies to the Session ID option. If it is present in the Server Response, then it MUST be present with the exact same option value in the following Echo Request messages. "RQ" means that the server SHOULD include the option in the response, when requested by the client using the Option Request option.

3.5. Rate Limiting

Clients MUST by default send at most Default-Client-Request-Rate (Section 3.5.1) Echo Request messages per second. Note that the value can be less than 1. Servers MUST by default perform rate limiting, to guard against this protocol being used for denial-of-service (DoS) attacks. A server MUST by default limit the number of clients that can be served at the same time, and for a given client, a server MUST also by default respond to, on average, at most Default-Server-Rate-Limit (see Section 3.5.1) Echo Request messages per second. Note that the value can be less than 1. Server implementations should provide configuration options allowing certain clients to perform more rapid rates of Echo Request messages. If higher rates are allowed for specific client IP addresses, then Init messages and the Session ID option MUST be used to help mitigate spoofing.

Implementers of applications/tools using this protocol **SHOULD** consider the UDP guidelines [RFC5405], in particular if clients are to send, or servers are to accept, Echo Request messages at rates exceeding the defaults given in this document. See Section 8, "Security Considerations", for additional discussion.

3.5.1. Message Rate Variables

There are two variables that control message rates. They are defined as follows.

Default-Client-Request-Rate

This variable defines the default client echo request rate, specifying the number of requests per second. Note that the value may be less than one. For example, a value of 0.1 means one packet per 10 seconds. The value 1 is **RECOMMENDED**, but the value might be too small or large, depending on the type of network in which the client is deployed. The value 1 is chosen because it should be safe in most deployments, and it is similar to what is typically used for the common tool "ping" for ICMP Echo Request messages.

Default-Server-Rate-Limit

This variable defines the default per-client rate limit that a server uses for responding to Echo Request messages. The average rate of replies **MUST NOT** exceed Default-Server-Rate-Limit per second. Note that the value may be less than one. For example, a value of 0.1 means an average of one packet per 10 seconds. The value 1 is **RECOMMENDED**, but the value might be too small or large, depending on the type of network in which the client is deployed. The value 1 is chosen because it should be safe in most deployments. This value **SHOULD** be high enough to accept the value chosen for the Default-Client-Request-Rate.

4. Client Behaviour

We will consider how a typical interactive client using the above protocol would behave.

A client only requires a user to specify the unicast address of the server. It can then send an Init message with a prefix option containing the desired address family and zero prefix length (wildcard entry). The server can then decide which group, from the specified family, it should return. A client may also allow the user to specify group address(es) or prefix(es) (for IPv6, the user may

only be required to specify a scope or a Rendezvous Point (RP) address, from which the client can construct the desired prefix, possibly embedded-RP). From this, the client can specify one or more prefix options in an Init message to tell the server which address it would prefer. If the user specifies a group address, that can be encoded as a prefix of maximal length (e.g., 32 for IPv4). The prefix options are in prioritised order; i.e., the client should put the most preferred prefix first.

If the client receives a Server Response message containing a group address, it can start sending Echo Request messages; see the next paragraph. If there is no group address option, the client would typically exit with an error message. The server may have included some prefix options in the Server Response. The client may use this to provide the user some feedback on what prefixes or scopes are available.

Assuming the client got a group address in a Server Response, it can start the multicast pings, after letting the user know which group is being used. Normally, a client should send at most Default-Client-Request-Rate (Section 3.5.1) Echo Request messages per second.

When sending the Echo Request messages, the client must always include the group option. If the Server Response contained a Session ID option, then it must also include a Session ID option, with the exact same value, in the Echo Request messages. If a client receives a Server Response message in response to an Echo Request (that is, a Server Response message containing a sequence number), this means there is an error, and it should stop sending Echo Request messages. This could happen after server restart.

The client may allow the user to request server information. If the user requests server information, the client can send an Init message with no prefix options, but with an Option Request option, requesting that the server return a Server Information option. The server will return server information, if supported, and it may also return a list of prefixes it supports. It will not, however, return a group address. The client may also try to obtain only a list of prefixes by sending an Init message with no prefixes and not requesting any specific options.

Although this technique is not recommended, a client may pick a multicast group and send Echo Request messages without first going through the Init - Server Response negotiation. If this is supported

by the server and the server is okay with the group used, the server can then send Echo Reply messages as usual. If the server is not okay with the group used, it will send a Server Response telling the client to stop.

5. Server Behaviour

We will consider how a typical server using the above protocol would behave, first looking at how to respond to Init messages.

If the Init message contains prefix options, the server should look at them in order and see if it can assign a multicast address from the given prefix. The server would be configured with a (possibly default) set of groups it can offer. It may have a large pool and pick a group at random, or possibly choose a group based on hashing of the client's IP address or identifier, or simply use a fixed group. A server could possibly decide whether to include site-scoped group ranges based on the client's IP address. It is left to the server to decide whether it should allow the same address to be used simultaneously by multiple clients.

If the server finds a suitable group address, it returns this address in a group option in a Server Response message. The server should additionally include a Session ID. This may help the server if it is to keep some state -- for instance, to make sure the client uses the group assigned to it. A good Session ID would be a pseudo-random byte string that is hard to predict; see [RFC4086]. If the server cannot find a suitable group address, or if there were no prefixes in the Init message, it may send a Server Response message containing prefix options listing what prefixes may be available to the client. Finally, if the Init message requests the Server Information option, the server should include that option.

When the server receives an Echo Request message, it must first check that the group address and Session ID (if provided) are valid. If the server is satisfied, it will send a unicast Echo Reply message back to the client, and also a multicast Echo Reply message to the group address. The Echo Reply messages contain the exact options (but no Session ID), and in the same order as in the Echo Request; after that, the server adds a TTL option and additional options if needed. For example, it may add a timestamp if requested by the client. If the server is not happy with the Echo Request (such as bad group address or Session ID, or request is too large), it may send a Server Response message asking the client to stop. This Server Response must echo the sequence number from the Echo Request.

This Server Response may contain group prefixes from which a client can try to request a group address. The unicast and multicast Echo Reply messages have identical UDP payload, apart from possibly TTL and timestamp option values.

Note that the server may receive Echo Request messages with no prior Init message. This may happen when the server restarts or if a client sends an Echo Request with no prior Init message. The server may go ahead and respond if it is okay with the group and Session ID (if included) used. If it is not okay with this information, the server sends back a Server Response.

6. Recommendations for Implementers

The protocol, as specified, is fairly flexible and leaves a lot of freedom for implementers. In this section, we present some recommendations.

Server administrators should be able to configure one group prefix or multiple group prefixes in a server implementation. When deploying servers on the Internet and in other environments, the server administrator should be able to restrict the server to respond to only a few multicast groups that should not be currently used by multicast applications. A server implementation should also provide flexibility for an administrator to apply various policies to provide one group prefix or multiple group prefixes to specific clients, e.g., site-scoped addresses for clients that are inside the site.

As specified in Section 3.5, for a given client, a server must by default respond to at most an average rate of Default-Server-Rate-Limit Echo Request messages per second. A leaky bucket algorithm is suggested, where the rate can be higher for a few seconds, but the average rate should by default be limited to Default-Server-Rate-Limit messages per client per second. Server implementations should provide administrative control of which client IP addresses to serve, and may also allow certain clients to perform more rapid rates of Echo Request messages.

If a server uses different policies for different IP addresses, it should require clients to send Init messages and return an unpredictable Session ID to help mitigate spoofing. This is an absolute requirement if exceeding the default rate limit. See the specification in Section 3.5.

7. IANA Considerations

IANA has assigned UDP user port 9903 (multicast-ping) for use by this protocol. IANA also provides registries for message and option types.

IANA has created a message types registry. Message types are in the range 0-255. Message types 0-253 are registered following the procedures for Specification Required from RFC 5226 [RFC5226], while types 254 and 255 are for experimental use. The registry includes the messages defined in Section 3.4. A message specification **MUST** describe the behaviour with known option types as well as the default behaviour with unknown option types.

IANA has created an option type registry. Option types 0-65531 are registered following the procedures for Specification Required from RFC 5226 [RFC5226], while types 65532-65535 are for experimental use. The registry should include the options defined in Section 3.2. An option specification must describe how the option may be used with the known message types. This includes which message types the option may be used with.

The initial registry definitions are as follows:

Multicast Ping Protocol Parameters:

Registry Name: Multicast Ping Protocol Message Types

Reference: RFC 6450

Registration Procedures: Specification Required

Registry: Type	Name	Reference
-----	-----	-----
65	Echo Reply	RFC 6450
73	Init	RFC 6450
81	Echo Request	RFC 6450
83	Server Response	RFC 6450
254-255	Experimental	

Registry Name: Multicast Ping Protocol Option Types
 Reference: RFC 6450
 Registration Procedures: Specification Required

Registry: Type	Name	Reference
0	Version	RFC 6450
1	Client ID	RFC 6450
2	Sequence Number	RFC 6450
3	Client Timestamp	RFC 6450
4	Multicast Group	RFC 6450
5	Option Request	RFC 6450
6	Server Information	RFC 6450
7	Deprecated	RFC 6450
8	Deprecated	RFC 6450
9	TTL	RFC 6450
10	Multicast Prefix	RFC 6450
11	Session ID	RFC 6450
12	Server Timestamp	RFC 6450
65532-65535	Experimental	

8. Security Considerations

There are some security issues to consider. One is that a host may send an Echo Request with an IP source address of another host, and make an arbitrary multicast ping server on the Internet send packets to this other host. This behaviour is fairly harmless. The worst case is if the host receiving the unicast Echo Reply messages also happens to be joined to the multicast group used. This is less of a problem for SSM, where also the source address of the server must match the address joined. In this case, there would be an amplification effect, where the host receives twice as many replies as there are requests sent. See below for how spoofing can be mitigated.

For ASM (Any-Source Multicast), a host could also make a multicast ping server send multicast packets to a group that is used for something else, possibly disturbing other uses of that group. However, server implementations should allow administrators to restrict which groups a server responds to. The administrator should then try to configure a set of groups that are not used for other purposes. Another concern is bandwidth. To limit the bandwidth used, a server **MUST** by default limit the number of clients that can be served at the same time, and a server **MUST** also by default perform per-client rate limiting.

In order to help mitigate spoofing, a server **SHOULD** require that the client send an Init message, and return an unpredictable Session ID in the response. The ID should be associated with the IP address and have a limited lifetime. The server **SHOULD** then only respond to Echo Request messages that have a valid Session ID associated with the source IP address of the Echo Request. Note, however, that a server is replying with a Server Response message if the Session ID is invalid. This is used to tell the client that something is wrong and that it should stop sending requests, and start over if necessary. This means, however, that someone may spoof a client request, and have the server send a message back to the client address. One solution here would be for the server to have a very low rate limit for the Server Response messages.

Note that the use of a Session ID only to some degree helps mitigate spoofing. An attacker that is on the path between a client and a server may eavesdrop the traffic, learn a valid Session ID, and generate Echo Request messages using this ID. The server will respond as long as the Session ID remains valid.

This protocol may be used to establish a covert channel between a multicast ping client and other hosts listening to a multicast group. A client can, for instance, send an Echo Request containing an undefined option with arbitrary data. The server would echo this back in an Echo Reply that may reach other hosts listening to that group. One solution that should be considered for future protocol versions is to reply with a hash of the data, rather than simply a copy of the same data.

9. Acknowledgments

The ssmping concept was proposed by Pavan Namburi, Kamil Sarac, and Kevin C. Almeroth in the paper "SSM-Ping: A Ping Utility for Source Specific Multicast" (2004) and also "MPing: A Ping Utility for IP Multicast" (Sarac and Almeroth, 2004). Mickael Hoerdet contributed several ideas. Alexander Gall, Nicholas Humfrey, Nick Lamb, and Dave Thaler have contributed in different ways to the implementation of the ssmping tools [IMPL]. Many people in communities like the Trans-European Research and Education Networking Association (TERENA), Internet2, and the M6Bone (IPv6 multicast network) have used early implementations of ssmping and provided feedback that influenced the current protocol. Thanks to Kevin Almeroth, Tony Ballardie, Bill Cervený, Toerless Eckert, Marshall Eubanks, Gorry Fairhurst, Alfred Hoenes, Liu Hui, Bharat Joshi, Olav Kvittem, Hugo Santos, Kamil Sarac, Pekka Savola, Trond Skjesol, and Cao Wei for reviewing and providing feedback on this document. In particular, Hugo, Gorry, and Bharat provided lots of input on several revisions of the document.

10. References

10.1. Normative References

[ADDRFAMILY]

IANA, "Address Family Numbers",
<<http://www.iana.org/assignments/address-family-numbers>>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

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

10.2. Informative References

[IMPL] Venaas, S., "ssmping implementation",
<<http://software.uninett.no/ssmping/>>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

Author's Address

**Stig Venaas
Cisco Systems
Tasman Drive
San Jose, CA 95134
USA**

EMail: stig@cisco.com