Network Working Group	S. Venaas
Internet-Draft	UNINETT
Intended status: Standards Track	November 03, 2008
Expires: May 7, 2009	

Multicast Ping Protocol
draft-ietf-mboned-ssmping-06

Status of this Memo

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 like multicast tree setup time etc. This protocol is based on an implementation of tools called ssmping and asmping.

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 (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.) [1].

Table of Contents

---

1. Introduction

TOC

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 like multicast tree setup time, the number of hops the packets have traveled, as well as packet delay and loss. This functionality resembles, in part, the ICMP Echo Request/Reply mechanism, but uses UDP RFC 768 (Postel, J., “User Datagram Protocol,” August 1980.) [2] and requires both a client and a server implementing this protocol. Intermediate routers are not required to support this protocol. They forward Protocol Messages and data traffic as usual.
The protocol here specified is based on the actual implementation of the ssmping and asmping tools [4] (, “ssmping implementation,” .) which are widely used by the Internet community to conduct multicast connectivity tests.

---

2. Architecture

TOC

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 typical protocol usage is as follows: A server runs continuously to

serve requests from clients. A client can test the multicast reception from this server, provided it knows a unicast address of the server. It will then send a unicast message to the server asking for a group to use. Optionally the user may have requested 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). Here G is the group specified by the server, and S is the unicast address used to reach the server. After joining the channel, the client unicasts multicast ping requests to the server. The requests are sent using UDP with destination port set to the standardised multicast ping port [TBD]. The requests are sent periodically, e.g., once per second, to the server. The requests contain a sequence number, and typically a timestamp. The requests are echoed by the server, except the server may add a few options. For each request, the server sends two replies. One reply is unicast back to the source IP address and source UDP port of the request, while another is multicast to the requested multicast group G and the source UDP port of the request. Both replies are sent from the same port on which the request was received. The server should specify the TTL used for both the unicast and multicast messages (we recommend at least 64) by including a TTL option; allowing the client to compute the number of hops. The client should leave the channel/group when it has finished its measurements.
By use of this protocol, a client can obtain information about several multicast delivery characteristics. First, by receiving unicast replies, it can verify that the server is receiving the unicast requests, 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; 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 (RTT). For unicast, the RTT is the time from when the unicast request is sent to when the reply is received. The measured multicast RTT also references the client's unicast request. By use of the TTL option 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 in the number of hops, RTT, etc. The number of multicast hops and changes in the number of hops over time, may also 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, and also differences in delay. Servers may optionally specify a timestamp. This may be useful since the unicast

and multicast replies can not be sent simultaneously (the delay depending on the host's operating system and load).

---

3. Protocol specification

TOC

There are four different message types. There are Echo Request and Echo Reply messages used for the actual measurements; there is an Init message that SHOULD be used to initialise a ping session and negotiate which group to use; and finally a Server Response message that 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. 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 generally 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. However, the server SHOULD add a TTL option, 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 Replies (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 that 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, and also 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

TOC

All options are TLVs formatted as specified 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Value                             |
|                               .                               |
|                               .                               |
|                               .                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type (2 octets) specifies the option. The different options are defined below.
Length (2 octets) specifies the length of the value field. Depending on the option type, it can be from 0 to 65535.
Value. The 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

TOC

This document defines the following options: Version (0), Client ID (1), Sequence Number (2), Client Timestamp (3), Multicast Group (4), Option Request Option (5), Server Information (6), TTL (9), Multicast Prefix (10), Session ID (11) and Server Timestamp (12). Values 7 and 8 are reserved. The options are defined below.

> Version, type 0
>
> > Length MUST be 1. This option MUST always be included in all messages, and the 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 Response message back with version set to 2. Client ID and Sequence Number options SHOULD be echoed if present. It SHOULD not include any other options. A client receiving a response with a version other than 2, MUST (unless it supports the particular version), stop sending requests to the server.

Client ID, type 1

> Length MUST be non-zero. A client SHOULD always include this option in all messages (both Init and Request). The client may use any value it likes to be able to detect whether a reply is a reply to its Init/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 Reply). The value might be a process ID, perhaps process ID combined with an IP address because it may receive multicast responses to queries from other clients. It is left to the client implementer how to make use of this option.

Sequence Number, type 2

> Length MUST be 4. A client MUST always include this in Request messages and MUST NOT include it in Init messages. A server replying to a Request message MUST copy it into the Reply (or Server Response message on error). This contains a 32 bit sequence number. The values would typically start at 1 and increase by one for each request in a sequence.

Client Timestamp, type 3

> Length MUST be 8 bytes. A client SHOULD include this in Request messages and MUST NOT include it in Init messages. A server replying to a Request message MUST copy it into the Reply. The timestamp specifies the time when the 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 last second since the Epoch.

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 Request messages. It MUST be used in 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 Reply messages (copied from the 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 [3] (, "IANA, Address Family Numbers," .). This is followed by the group address. Length of the option value will be 6 for IPv4, and 18 for IPv6.

Option Request Option, type 5

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

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Type          |          Option Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             .....                             |
```

This option might be used by the client to ask the server to include options like 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 Timestamp in 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 (Reply messages) to the 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 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.

Reserved, type 7

This option code value was used by early implementations for an option that is now deprecated. This option should no longer be used. Clients MUST NOT use this option. Servers MUST treat it as an unknown option (not process it if received, but if received in a Request message, it MUST be echoed in the Reply message).

Reserved, type 8

This option code value was used by early implementations for an option that is now deprecated. This option should no longer be used. Clients MUST NOT use this option. Servers MUST treat it as an unknown option (not process it if received, but if received in a Request message, it MUST be echoed in the Reply message).

TTL, type 9

Length MUST be 1. This option contains a single octet specifying the TTL of a Reply message. Every time a server sends a unicast or multicast 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 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.

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), it MAY be used in Server Response messages to tell the client what prefix(es) it may try to obtain a group from. It MUST NOT be used in 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 [3] (, “IANA, Address Family Numbers,” .). 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 non-zero. A server SHOULD include this in Server Response and Reply messages. If a client receives this option in a message, the client MUST echo the Session ID option in subsequent Request messages, with the exact same value, until the next message is received from the server. If the next message from the server has no Session ID or a new Session ID value, the client should do the same, either not use the Session ID, or use the new 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 chosen pseudo randomly so that it is hard to predict. This can be used to prevent spoofing of the source address of Request messages, see the Security Considerations for details.

Server Timestamp, type 12

Length MUST be 8 bytes. A server supporting this option, SHOULD include it in Reply messages, if requested by the client. The timestamp specifies the time when the 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 last second since the Epoch.

---

4. Packet Format

TOC

The format of all messages is a one octet message type, directly followed by a variable number of options.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |          Options ...                          |
+-+-+-+-+-+-+-+-+             .                                 |
|                             .                                 |
|                             .                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-      .....
```

There are four message types defined. Type 81 (the character Q in ASCII) specifies an Echo Request (Query). Type 65 (the character A in ASCII) specifies an Echo Reply (Answer). Type 73 (the character I in ASCII) is an Init message, and type 83 (the character S in ASCII) is a Server Response message.
The options directly follow the type octet and are not aligned in any way (no spacing or padding), i.e., options might start at any octet boundary. The option format is specified above.

5. Message types and options

TOC

There are four message types defined. We will now describe each of the message types and which options they may contain.

> Init, type 73
>
> > This message is sent by a client to request information from a server. It is mainly used for requesting a group address, but it may also be used to check which group prefixes the server may provide groups from, or other server information. It MUST include a Version option, and SHOULD include a Client ID. It MAY include Option Request and Multicast Prefix Options. This message is a request for a group address if and only if it contains Multicast Prefix options. If multiple Prefix options are included, they should be in prioritised order. I.e., the server will consider the prefixes in the order they are specified, and if it finds a group for a prefix, it will only return that one group, not considering the remaining prefixes.
>
> Server Response, type 83
>
> > This message is sent by a server. Either as a response to an Init, or in response to a Request. When responding to 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 a Request, the message tells the client to stop sending Requests. 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.

Echo Request, type 81

This message is sent by a client, asking the server to send unicast and multicast replies. It MUST include Version, Sequence Number and Multicast Group options. If the last message (if any) received from the server contained a Session ID, then this MUST also 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 echoes most of the options from the Echo Request (any options in the Request that are unsupported by the server, are always echoed). The only option that may be present in the Request which is not always echoed, is the Session ID option. In most cases the server would echo it, but the server may also change or omit it. The two Reply messages SHOULD both contain a TTL option (not necessarily equal), and both SHOULD also contain Server Timestamps (not necessarily equal) when requested.

For the reader's convenience we provide the matrix below, showing what options can go in what messages.

Option / Message Type	Init	Server Response	Request	Reply
Version (0)	MUST	ECHO	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
Reserved (7)	NOT	NOT	NOT	ECHO
Reserved (8)	NOT	NOT	NOT	ECHO
TTL (9)	NOT	NOT	NOT	SHOULD
Multicast Prefix (10)	MAY	MAY	NOT	NOT
Session ID (11)	NOT	MAY	ECHO	MAY
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 means that it MUST echo the option it got in the last message from the server in any subsequent messages it sends. RQ means that the server SHOULD include the option in the response, when requested by the client using the Option Request option.

---

6. Client Behaviour

TOC

We will consider how a typical interactive client using the above protocol would behave. A client need only require 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 is then free to decide which group, from the specified family, it should return. A client may also allow a user to specify group address(es) or prefix(es) (for IPv6, the user may only be required to specify a scope or an 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 Request messages, see the next paragraph. If there is no group address option, it 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 pinging. Before it does that, it should let the user know which group is being used. Normally, a client should send at most one ping request per second. When sending ping Requests, the client must always include the group option. If the last message from the server contained a Session ID, then it must also include that with the same value. Typically it would receive a Session ID in a Server Response together with the group address, and then the ID would stay the same during the entire ping sequence. However, if for instance the server process is restarted, it may still be possible to continue pinging but the Session ID may be changed by the server. Hence a client implementation must always use the last Session ID it received, and not necessarily the one from the Server Response message. If a client receives a Server Response message in response to a Request message (that is, a Server Response message containing a sequence number), this means there is an error and it should stop sending Requests. This may for instance 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 the server to 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 however not 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. Note that a client may pick a multicast group and send 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 Reply messages as usual. If the server is not okay, it will send a Server Response telling the client to stop.

---

7. Server Behaviour

TOC

We will consider how a typical server using the above protocol would behave. First we consider 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, possibly have a default, specifying which groups it can offer. It may have a large pool just picking a group at random, possibly choose a group based on hashing of the client's IP address or identifier, or just 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 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 for making sure the client uses the group it got assigned. A good Session ID would be a pseudo random byte string that is hard to predict. 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, it should include that.
When the server receives a Request message, it may first check that the group address and Session ID (if provided) are valid. If the server is satisfied, it will send a unicast Reply message back to the client, and also a multicast Reply message to the group address. The Reply messages contain the exact options and in the same order, as in the Request, and after that the server adds a TTL option and additional options if needed. E.g., it may add a timestamp if requested by the client. If the server is not happy with the Request (bad group address or Session ID,

request is too large etc), it may send a Server Response message asking the client to stop. This Server Response must echo the sequence number from the Request. This Server Response may contain group prefixes from which a client can try to request a group address. The unicast and multicast Reply messages have identical UDP payload apart from possibly TTL and timestamp option values.

Note that the server may receive Request messages with no prior Init message. This may happen when the server restarts or if a client sends a Request with no prior Init message. The server may go ahead and respond if it is okay with the group used. In the responses it may add a Session ID which will then be in later requests from the client. If the group is not okay, the server sends back a Server Response. The Response is just as if it got an Init message with no prefixes. If the server adds or modifies the SessionID in replies, it must use the exact same SessionID in the unicast and multicast replies.

---

8. Recommendations for Implementers

TOC

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 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 which 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 or multiple group prefixes to specific clients, e.g., site scoped addresses for clients that are inside the site. Clients could be identified by their IP address provided that clients are required to send Init messages, and they receive an unpredictable Session ID. See also Section 11 (Security Considerations).

Clients should by default send at most one request per second. Servers should perform rate limiting, to guard against this protocol being used for DoS attacks. The server should for a given client, respond to at most one Request message 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 a message 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 requests. Implementers of applications/tools using this protocol should consider the UDP guidelines (Eggert, L. and G. Fairhurst, “Unicast UDP Usage Guidelines for Application Designers,” October 2008.) [5], in particular if clients are to send, or servers are to accept, requests at rates exceeding one per second. If higher rates are allowed for specific IP addresses, then Init messages and the

Session ID option should be used to help prevent spoofing. See Section 11 (Security Considerations).

---

9. Acknowledgments

TOC

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, and also the Internet Draft draft-sarac-mping-00.txt. Mickael Hoerdt has contributed with several ideas. Alexander Gall, Nicholas Humfrey, Nick Lamb and Dave Thaler have contributed in different ways to the implementation of the ssmping tools at [4] (, "ssmping implementation," .). Many people in communities like TERENA, Internet2 and the M6Bone have used early implementations of ssmping and provided feedback that have influenced the current protocol. Thanks to Kevin Almeroth, Toerless Eckert, 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 draft. In particular Hugo, Gorry and Bharat have provided lots of input on several revisions of the draft.

---

10. IANA Considerations

TOC

IANA is requested to provide a well-known UDP port number for use by this protocol, and also to provide registries for message and option types.

There should be a message types registry. Message types are in the range 0-255. Message types 0-191 require specification (an RFC or other permanent and readily available reference), while types 192-255 are for experimental use and are not registered. The registry should include the messages defined in Section 5 (Message types and options). A message specification must describe the behaviour with known option types as well as the default behaviour with unknown ones.

There should also be an option type registry. Option types 0-49151 require specification (an RFC or other permanent and readily available reference), while types 49152-65535 are for experimental use and are not registered. The registry should include the options defined in Section 3.2 (Defined Options). 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 could be as follows:

Multicast Ping Protocol Parameters:

Registry Name: Multicast Ping Protocol Message Types
Reference: [this doc]
Registration Procedures: Specification Required

Registry:

Type	Name	Reference
65	Echo Reply	[this doc]
73	Init	[this doc]
81	Echo Request	[this doc]
83	Server Response	[this doc]
192-255	Experimental	

Registry Name: Multicast Ping Protocol Option Types
Reference: [this doc]
Registration Procedures: Specification Required

Registry:

Type	Name	Reference
0	Version	[this doc]
1	Client ID	[this doc]
2	Sequence Number	[this doc]
3	Client Timestamp	[this doc]
4	Multicast Group	[this doc]
5	Option Request Option	[this doc]
6	Server Information	[this doc]
7	Reserved	[this doc]
8	Reserved	[this doc]
9	TTL	[this doc]
10	Multicast Prefix	[this doc]
11	Session ID	[this doc]
12	Server Timestamp	[this doc]
49152-65535	Experimental	

11. Security Considerations

TOC

There are some security issues to consider. One is that a host may send a 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 replies also happen to be joined to the multicast group used. 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 prevented.

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. The main concern is bandwidth. Since there is a well-known port, it should not be received by other applications. Due to this, a server on the Internet SHOULD perform rate limiting.

In order to help prevent spoofing, a server SHOULD require the client to 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 Request messages that have a valid Session ID associated with the source IP address of the Request.

Server implementations should allow administrators to restrict which groups a server responds to, and also perform rate limiting. This is discussed in Section 8 (Recommendations for Implementers).

---

12. References

TOC

12.1. Normative References

TOC

[1]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[2]	Postel, J., "User Datagram Protocol," STD 6, RFC 768, August 1980 (TXT).
[3]	"IANA, Address Family Numbers."

12.2. Informative References

TOC

[4]	"ssmping implementation."
[5]	Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers," draft-ietf-tsvwg-udp-guidelines-11 (work in progress), October 2008 (TXT).

Author's Address

TOC

	Stig Venaas
	UNINETT
	Trondheim NO-7465

	Norway
Email:	venaas@uninett.no

Full Copyright Statement

TOC

Intellectual Property