

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 10, 2008

S. Venaas
UNINETT
H. Santos
July 9, 2007

ssmping Protocol
draft-ietf-mboned-ssmping-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 10, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

ssmping is a tool that is used to check whether one can receive SSM, as well as obtaining some additional information. ssmping requires both a client and a server supporting the ssmping protocol to work. We here specify this protocol.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

Internet-Draft

ssmping

July 2007

"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [1].

Table of Contents

1.	Introduction	3
2.	Architecture	3
3.	Protocol specification	4
3.1.	Option format	5
3.2.	Defined Options	5
4.	Packet Format	7
5.	Acknowledgements	8
6.	IANA Considerations	8
7.	Security Considerations	8
8.	References	9
8.1.	Normative References	9
8.2.	Informative References	9
	Authors' Addresses	9
	Intellectual Property and Copyright Statements	10

Internet-Draft

ssmping

July 2007

1. Introduction

ssmping is a tool that is designed to allow a local host to check whether it is able to receive a multicast flow (SSM by default, or ASM when specific options are used) originated by a remote host. Additionally it is able to report other information such as the amount of time used to establish the multicast tree, the number of hops the flow's packets have traveled as well as the packet delay and loss. This functionality resembles in part the ICMP Echo Request/Reply infrastructure but over UDP and implemented by both the ssmping client and server. The protocol here specified is based on the actual implementation of the ssmping tool [3] which is widely used by the Internet community to conduct multicast connectivity tests.

2. Architecture

Before going into the protocol details we will describe how it is used and what information it may provide. The typical usage of an ssmping session is as follows. A server runs continuously in order to serve request from clients. When a host decides to verify the multicast reception from a specific server (knowing one of the server's unicast addresses is required), the ssmping client joins an SSM channel (S,G) where S is a unicast address of the target server and G is the standard multicast group defined for use by ssmping.

After joining the channel, the client sends ssmping requests encapsulated in UDP to the standardised ssmping port and the unicast address of the server. The requests are sent periodically, e.g. once per second, to the server. The requests contain a serial number, and typically a timestamp. The requests are typically, but not necessarily always, simply echoed back by the server. To each request, the server sends two replies. One as unicast back to the port and address the request was sourced from, and also as multicast back to the port the request came from. It is currently left open which port the request is sourced from, whether this port should be

standardised or not. The TTL or Hop Limit of the replies are set to 64. The client should leave the SSM channel when it has finished its measurements.

By use of this protocol, a client can obtain information on several aspects of the multicast quality. First of all, 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 in receiving 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 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 the unicast request is sent to when the reply is received. The measured multicast RTT also references the client's unicast request. Since the server sets TTL or Hop Limit to 64, it can also know the number of router hops it is away from the source. By obtaining the same values by 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. Provided that the server sends the unicast and multicast replies nearly simultaneously, it may also be able to measure difference in one way delay for unicast and multicast on the path from server to client, and also differences in delay variation. 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), or when the client and server have synchronised clocks.

[3.](#) Protocol specification

The ssmping requests and replies have 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. Generally the client includes a number of options in the request, and a server may simply echo the content back (only changing the message type), without inspecting the options. However, there are a number of options that a server implementation may support, where the client may ask for a certain information or behaviour from

the server. In some cases the server will need to add options in the response. The response will then first contain the exact options from the request, and then right after those, options appended by the server.

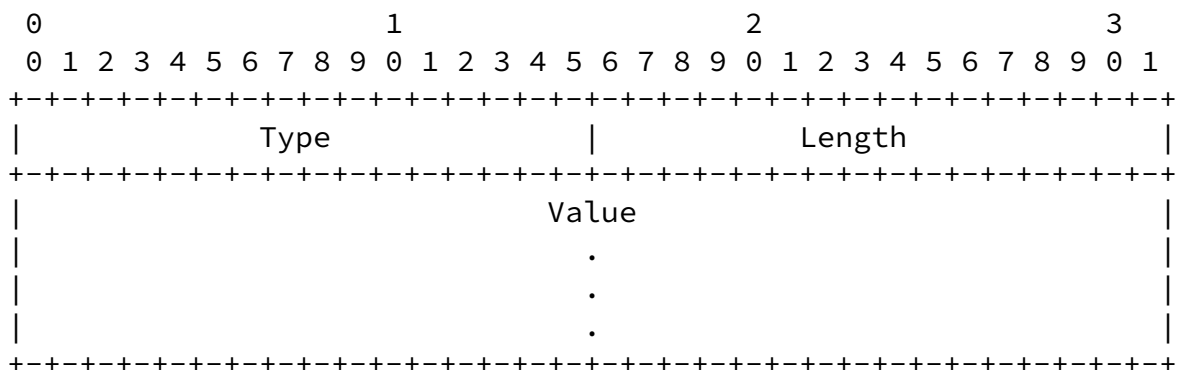
This document defines a number of different options. Some options don't require processing by servers and are simply returned unmodified in the reply. There are however other client options that the server may care about, and also server options that may be requested by a client. Generally a simple client will only include a few options, and get exactly the same options and values echoed back. Strictly speaking the protocol could work without any options. The protocol here defined does not require the use of any options, and a client may operate without specifying any. However some of the options allow the client to obtain additional information.

Unless otherwise specified, an option **MUST NOT** be used multiple times in a request. Also unless otherwise specified, an option **MUST NOT** be appended by the server multiple times. Note that some options, like

timestamp, may be added by both the client and the server. In that case the timestamp option would be in the response twice. But as said above, it is not used multiple times in the request, and not appended multiple times by the server.

[3.1.](#) Option format

All options are TLVs formatted as specified below.



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

Client Identifier, type 1. Length MUST be non-zero. Only used by clients. A client SHOULD include this. The client may use any value it likes to be able to detect whether a reply is a reply to this query or not. A server should treat this as opaque data, and simply leave it unchanged in the reply. The value might be a process ID, perhaps process ID combined with an IP address because it may receive multicasted responses to queries from other clients. It is left to the client implementor how to make use of this.

Sequence number, type 2. Length MUST be 4. Only used by clients. A client SHOULD include this. This contains a 32 bit sequence number. The values would typically start at 1 and increase by one for each request in a sequence.

Timestamp, type 3. Length MUST be 8 bytes. A client SHOULD include this. A server MAY support this. If supported it SHOULD be included

in the reply if requested by the client. The timestamp specifies the time when the message (query or reply) is sent. The first 4 bytes specify the number of seconds since the Epoch (beginning of the year 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 1. It is optional for clients and servers to support this. It allows a client to specify which group the server should send to. This is currently used by a tool called "asmping" to test ASM connectivity. The server may have restrictions on which groups can be used. 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			

```

+++++
| Addr Family | Multicast group address... |
+++++

```

The address family is a value 0-127 as assigned by IANA for Internet Address Families [2]. This is followed by the group address. For IPv4 the option value length will be 5, for IPv6 17.

Option Request Option, type 5. Length MUST be greater than 1. The option contains a list of option types of options that the client requests from the server. Supporting this is optional for both clients and servers. The length of this option will be a non-zero even number, since it contains option types that each are two octets. The format of the 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          |
+++++
|                               | ..... |

```

The value might contain an odd number of options, including just one. This option might be used by the client to ask the server to include options like timestamp or version.

Version, type 6. Length MUST be non-zero. Supporting this option is optional. A server supporting this option SHOULD add it if and only if requested by the client. The value is just unformatted text that might contain vendor and version information for the server implementation. It may also contain information on which options the

server supports.

Type 7, Reserved. This option code value was used by early implementations for an option that now is deprecated. This should no longer be used. Clients MUST not use this option, and Servers MUST ignore it.

Pad, type 8. Length can be anything, including 0. This option is

any way (no spacing or padding). I.e., options might start at any octet boundary. The option format is specified above.

5. Acknowledgements

The ssm ping 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 Hoerdts has contributed with several ideas. Alexander Gall, Nicholas Humfrey, Nick Lamb and Dave Thaler have contributed in different ways to my implementation of the ssm ping tools [3]. Many people in communities like TERENA, Internet2 and the M6Bone have used early implementations of ssm ping and provided feedback that have influenced the current protocol. Thanks to Kevin Almeroth, Toerless Eckert, Gorrry Fairhurst, Liu Hui, Olav Kvittum, Kamil Sarac, Pekka Savola, Trond Skjesol and Cao Wei for reviewing and providing feedback on this draft.

6. IANA Considerations

As currently specified, ssm ping would need a well known port number which the servers listen to. It might be desirable to use SRV records instead or in addition to this. For IPv6 SSM ssm ping should ideally have a reserved group ID. For the optional ASM functionality it would be useful to have a reserved IPv6 group ID, this may be the same as the one used for SSM. It may also be useful to have a dedicated group for the optional IPv4 ASM functionality. This section needs further work.

There may also be a need for an ssm ping option registry. The exact IANA considerations need to be clarified before this document can go to working group last call.

7. Security Considerations

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 a random ssm ping server on the Internet send packets to this other host. This is fairly harmless. The worst case is if the host receiving the unicast replies also happen to be performing an ssm ping test towards that particular server. In this unlikely event there would be an amplification effect where the host receives twice as many replies as there are requests sent. An ssm ping server should perform rate limiting, to guard against this being used as a DoS

attack. A client should also use the client identifier option to be able to distinguish replies to its own requests from replies that might be to other requests. How the protocol should be designed to cope with rate limiting at the server requires further study. One possibility might be that the server can choose to send generic replies, e.g. a packet every second without the usual client options but including sequence number and server time stamp, and where clients do not send requests as long as they receive generic replies.

8. References

8.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [2] "IANA, Address Family Numbers",
<<http://www.iana.org/assignments/address-family-numbers>>.

8.2. Informative References

- [3] "ssmping implementation",
<<http://www.venaas.no/multicast/ssmping/>>.

Authors' Addresses

Stig Venaas
UNINETT
Trondheim NO-7465
Norway

Email: venaas@uninett.no

Hugo Santos
Urb Glicinias, Smart Residence, 211
Aveiro 3810
Portugal

Email: hugo@fivebits.net

Internet-Draft

ssmping

July 2007

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at

ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

Venaas & Santos

Expires January 10, 2008

[Page 10]