Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 30, 2007

S. Venaas
UNINETT

February 26, 2007

ssmping Protocol
draft-venaas-mboned-ssmping-00

Status of this Memo

Copyright Notice

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",

"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

ssmping is a tool that is used to check whether one can receive SSM, and it can also give other information like the time to establish the tree, number of router hops the packets have traveled, packet delay and loss. The ssmping functionality resembles ICMP echo request/ reply using UDP and a client and a server that supports the ssmping protocol. It is used by a client to verify that it can receive multicast from the server, as well as some additional information. The protocol as specified here is based on an actual implementation of a tool [3] that has been found useful by many organisations.

2. Protocol description

Before going into the protocol details we will describe how it is used and what information it may provide. The typical usage is as follows. A server runs continuously in order to serve request from clients. At some point a client application may try to verify multicast reception from such a server. The client will need to know a unicast address of a server. The client joins an SSM channel (S,G) where S is a unicast address of the server, and G is a standardised multicast group for use by ssmping. After joining the channel, the client sends ssmping requests as UDP to a 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 is indeed caused by a multicast problem. If it does receive multicast, it knows not only that it can receive, but it may get some information on how long it takes 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 unicast request is sent to when the reply is received. For multicast we also talk about RTT, but then we

mean from the unicast request is sent to when the multicast reply is received. 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 comparing with the unicast replies, it can see whether there are differences in RTT and number of hops etc for unicast and multicast. 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 if there are differences in delay variation. Servers may optionally specify a timestamp. This may be useful if the unicast and multicast replies can not be sent nearly simultaneously, or if the client and server have synchronised clocks.

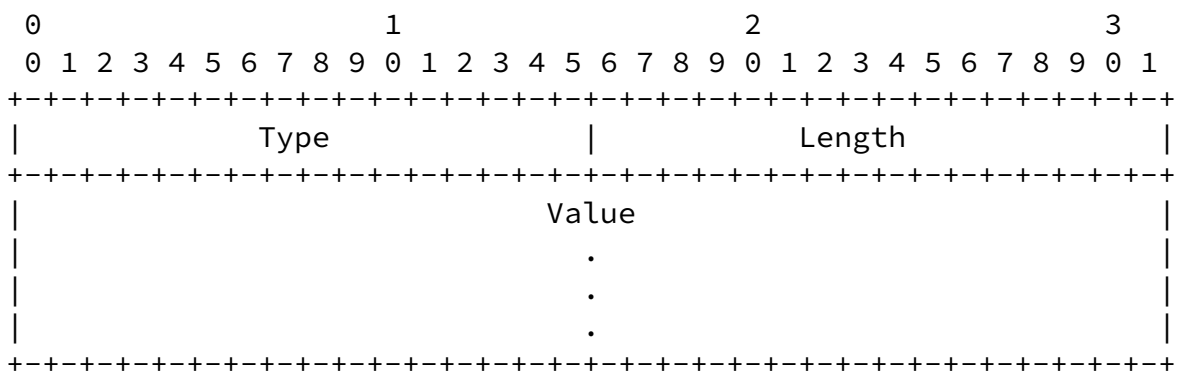
The ssmping requests and replies have a common format, one octet specifying the message type, followed by a number of options in TLV (Type, Length and Value) format. This makes the protocol easily extendible. Generally the client includes a number of options in the request, and a server may simply echo 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.

3. Options

There are a number of different options. Most of the options are only used by clients and simply echoed back by the server, where the server doesn't care about their contents. There are however some client options that the server may care about. There are also server options that may be requested by the 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. Without sending any options a client would still be able to tell whether multicast is working or not, however with the use of some of the basic options a client can obtain a lot more information.

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. 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

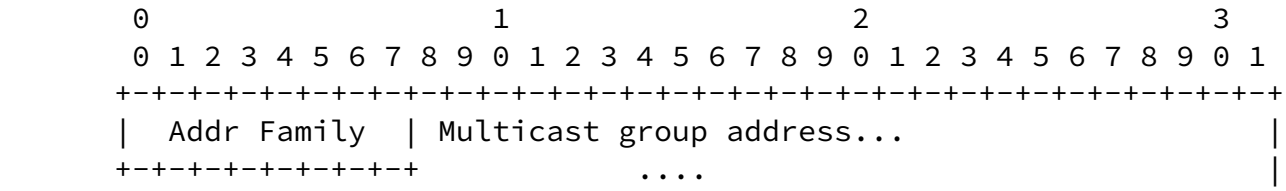
Client Identifier, type 1. Length MUST be non-zero. Only used by clients. A client SHOULD include this. The client may use any value it likes to be able to detect whether a reply is a reply to this query or not. A server should treat this as opaque data, and simply leave it unchanged in the reply. The 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.

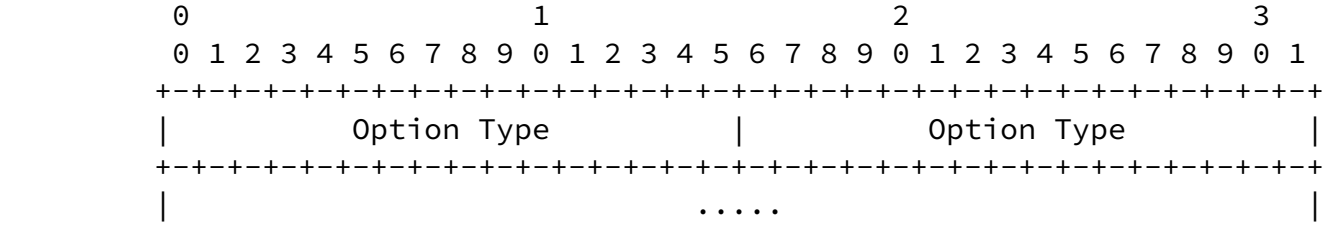
Multicast group, type 4. Length MUST be greater than 1. It is optional for clients and servers to support this. It allows a client to specify which group the server should send to. This is currently

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.



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.



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.

Reply size, type 7. Length MUST be 2 octets. This option is optional for clients and servers. It can be used to request the server response to be of a certain size. The value specifies the desired response size in octets. A server supporting this will if necessary use the pad option to increase the size of the response. A server should however not try to make the response shorter due to this option. That is, it should not omit or shorten any option

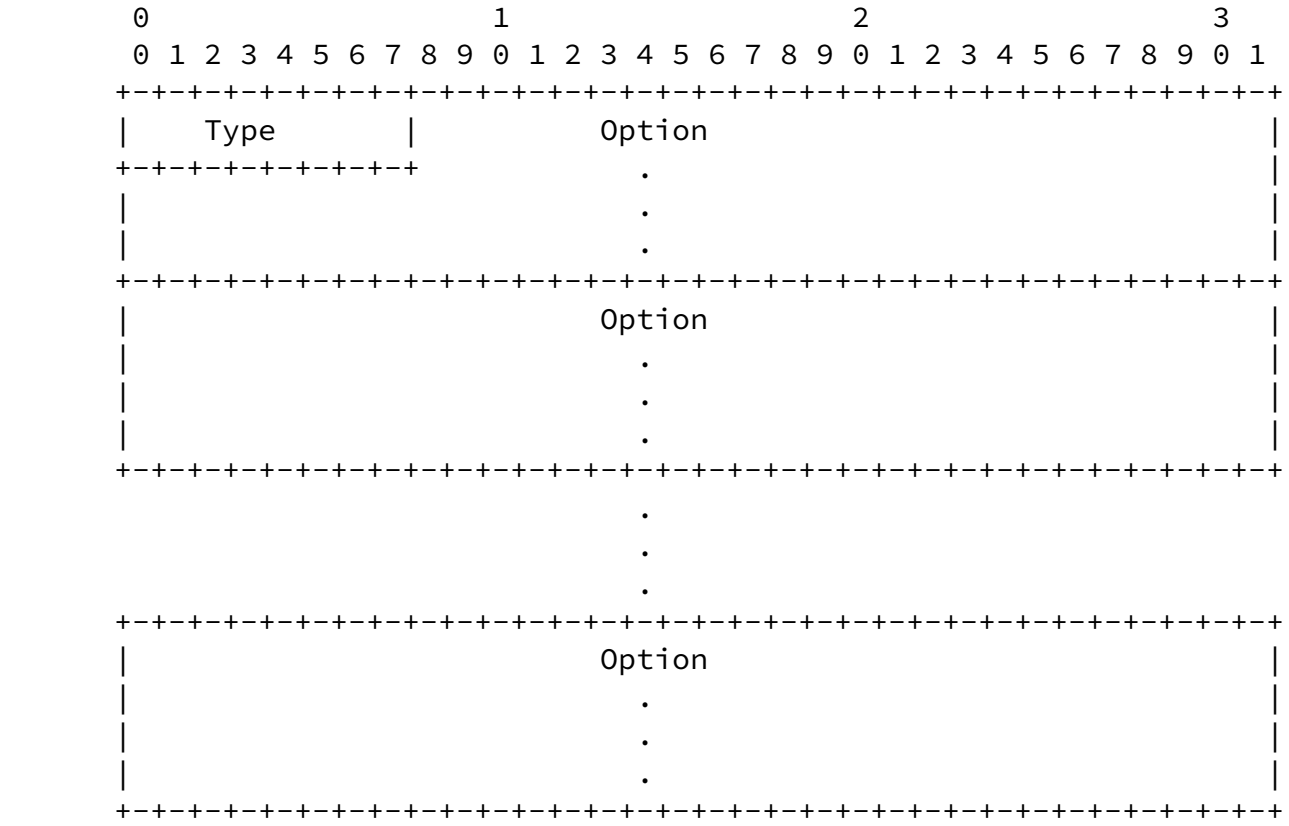
values to try to accommodate this. The response should never be shorter than if this option were not included. Also, the pad option requires at least 3 octets, so the server will not pad the response size if the requested size is not at least 3 octets longer than the normal response size.

Pad, type 8. Length can be anything, including 0. This option is used by servers to increase the response size if the client asks for

a reply that is larger than what the server normally would send. The addition of this option consumes a minimum of 3 octets, so it should only be added if the requested size is at least 3 octets more than the size of the normal (non-padded) response.

4. Packet Format

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



There are two message types defined. Type 81 (the character Q in ASCII) specifies a query. Type 65 (the character A in ASCII) specifies a response (answer).

The options follow right after 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 below

5. Acknowledgements

The ssmping idea 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, Nick Lamb and Dave Thaler have contributed in different ways to my implementation of the ssmping tools [3]. Hugo Santos has made an independent implementation of an ssmping server. 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 Olav Kvittem, Kamil Sarac and Trond Skjesol for reviewing and providing feedback on this draft.

6. IANA Considerations

As currently specified, ssmping 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 ssmping 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.

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 ssmping 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 ssmping 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 ssmping server should perform rate limiting, to guard against this being used as an 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/>.

Author's Address

Stig Venaas
UNINETT
Trondheim NO-7465
Norway

Email: venaas@uninett.no

Full Copyright Statement

Intellectual Property

ietf-ipr@ietf.org.

Acknowledgment

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