Abstract

When implementing signaling protocols such as GIMPS, implementors need a way to test the functionality and measure the performance of their own implementations. In this document, we try to provide a sketch for such a testing tool, a simple, stateless Ping Protocol, which works similar to ICMP Ping. This tool is able to traverse a path from a source to a destination along signaling aware network
nodes and collect various data that could be useful for identifying each node it is passing.

Table of Contents

1. Introduction .................................................. 3
2. Design Overview ............................................. 3
   2.1 Ping message format ..................................... 5
   2.2 Behaviour of nodes running the Ping tool .............. 7
3. Possible extension to the current ping functionality .... 7
4. Summary and Open Issues ................................. 8
5. Security Considerations ................................. 8
6. Acknowledgments ............................................. 8
7. References .................................................. 8
   7.1 Normative References ................................... 8
   7.2 Informative References ................................. 9
   Authors’ Addresses .......................................... 9
   Intellectual Property and Copyright Statements .......... 10
1. Introduction

This document describes a design for the implementation of a simple and basic stateless Ping tool for traversal of General Internet Messaging Protocol for Signaling (GIMPS) [1] aware network nodes.

In the NSIS two-layer architecture, GIMPS is being developed as the fundamental building block to provide generic signaling services for various signaling applications. Without implementing any full-fledged signaling application, GIMPS implementors may want to test the functionality and run-time properties of the protocol. A tool for such purposes, so-called "Ping Tool" in the document, which is inspired by the ping client done in the implementation of the Cross Application Signaling Protocol (CASP) [2] at Univ Goettingen, suffices this need.

An implementation of the ping tool is able to traverse each GIMPS aware node from initiator to responder and back to the initiator. Useful information about the signaling behaviour e.g., information about the signaling-aware hops and GIMPS layer processing delays is collected while traversing the network.

The initial functionality of such a Ping tool would be rather simple; details will be described later in this document. With this simplicity in mind, we reused the concept of the ’Null Service Type’ as described in RFC2997 [3].

2. Design Overview

The design of the Ping tool should follow these basic rules:
- simplicity (with a minimal overhead)
- testing as many properties of GIMPS as possible

The ping tool proposed in this draft uses the layered structure of NSIS, and is defined as a simple NSIS signaling Layer Protocol (NSLP) application. The ping tool uses the common API to communicate with the NSIS transport Layer Protocol (NTLP) and so it is able to test the functionality of GIMPS from the NSLPs’ point of view.

The ping tool consists of two parts. The ‘Ping Daemon’ is the NSLP application that does the real work of sending and receiving ping messages. The ‘Ping Client’ as a user side program is used to trigger the ‘Ping Daemon’ in a GIMPS node to send ping messages. Additionally the ‘Ping Client’ can perform the anlysis of the collected data.

Figure 1 shows the layering of the ping tool and the common packet flow provided by GIMPS, where the Initiator sends data packets along
the path through GIMPS-aware nodes until they reach the Responder. This Responder will send its response message upstream back to the Initiator.

![Diagram of Ping tool layering and packet flow overview](Image)

**Figure 1: Ping tool layering and packet flow overview**

The proposed ping tool uses the transport mechanisms provided by GIMPS. Unlike the end-to-end delivery provided by the IP, ping messages are sent hop-to-hop in GIMPS nodes. At each node running the "Ping Daemon", received ping messages are passed to the NSLP level, which decides which action should be taken next. Thus, Ping tool offers traceroute-like path discovery without adding any feature in GIMPS.

So the operation of the ping tool is as follows: The initiator sends a NSLP data message (using the ping message format described later) downstream towards the destination. The ping message is passed to each hop’s ‘Ping Daemon’, which will add the following information to the data message:

- Its own IP-address
- A timestamp with the current time since the Epoch (00:00:00 UTC, January 1, 1970) in microseconds.

When the ping message arrives at the receiver, the receiver adds its own information same as any other node and changes the direction from downstream to upstream. The nodes are passed in reverse order and again every hop adds its own information. The intermediate nodes do not change the sending direction of a ping message, so it finally arrives at the initiator. The collected data is passed to the ‘Ping Client’ which is able to calculate round trip times (RTTs) from the data collected along the path. Figure 2 shows a calculation example.
t1(0)  t1(1)  t1(2)  t1(3)         t1(N)
+---+  +---+  +---+  +---+         +---+
| I |>>| 1 |>>| 2 |>>| 3 |>> ... >>| R |
+---+  +---+  +---+  +---+         +---+
    v
+---+  +---+  +---+  +---+         +---+
| I |<<| 1 |<<| 2 |<<| 3 |<< ... <<| R |
+---+  +---+  +---+  +---+         +---+
t2(0)  t2(1)  t2(2)  t2(3)         t2(N)

\[ t_1(x) \text{ is the timestamp inserted by hop } x \text{ in downstream direction} \]
\[ t_2(x) \text{ is the timestamp inserted by hop } x \text{ in upstream direction} \]
where \( t_1(N) = t_2(N) \)

overall RTT for node \( x \) is: \( RTT(x) = t_2(x) - t_1(x) \)
hop-to-hop RTT for nodes \( x \) and \( y \) (\( x < y \)) can be computet by:
\[ h2hRTT(x, y) = RTT(x) - RTT(y) \]

Figure 2. An example of timestamp use

Note that the ‘Ping Daemon’ will not install any state in the NSLP level on the node it is running on, except for the initiator node. The Ping tool is therefore stateless. However the underlying GIMPS layer may, and probably will, install state according to GIMPS specifications, e.g., for reverse message routing.

2.1 Ping message format

The ping message format is used in downstream and upstream direction and is extended by every node on the path. As described above, currently two types of information are added to the Ping tool message by each node: IP-address and timestamp. Also, the number of hops, meaning the amount of nodes the packet traversed, and the message length should be present in such an message. Having both hop and length information adds redundancy but can help data analysis in the ping client. To support future extensions of the ping message format, a version number is attached. This draft represents version 1 in this context. Finally a sequence number is added to the ping message. This can be used to identify single ping messages if multiple pings are send concurrently. Figure 3 shows the ping message format in its final form when returning to the initiator. The IP-address and timestamp block for each hop are added to the message while traversing the GIMPS network.
Figure 3. Ping message format
Note that for compatibility with IPv4 and IPv6 the size of each IP-address field will be 16 bytes. The timestamp uses 4 bytes for seconds since Epoch (00:00:00 UTC, January 1, 1970) and additional 4 bytes for microseconds. This example shows that each hop, except the Nth one, adds a timestamp twice, due to the fact that each hop is passed twice, one time in downstream and another time in upstream direction. Using this information, one can calculate round trip times (RTT) for every node very easily.

2.2 Behaviour of nodes running the Ping tool

There are four entities involved in a ping session. Detailed actions for each of those will be described here:

Behavoiour of ‘Ping Client’:
- Contact ‘Ping Daemon’ on local node
- Request sending ping message with specified receiver and sequence number
- Wait for response from ‘Ping Daemon’
- Process collected data and generate result output

Behavoiour of ‘Ping Daemon’ on the Initiator node:
- Create Ping message
- Add own IP-address and timestamp
- Send message downstream towards receiver
- Wait for message to return
- Pass message to the ‘Ping Client’ who requested the ping

Behavoiour of ‘Ping Daemon’ on intermediate nodes
- Receive Ping message
- Increase number of hops field by 1
- Add own IP-address and timestamp at the end of the message
- Adjust length field
- Forward message in the same direction it arrived

Behavoiour of ‘Ping Daemon’ on receiver node
- Receive Ping message
- Increase number of hops field by 1
- Add own IP-address and timestamp at the end of the message
- Adjust length field
- Send the message in upstream direction

3. Possible extension to the current ping functionality

Ping messages are currently only used for collecting hop and processing delay information. Further extensions for this ping tool are possible but require properly addressing security concerns:
Collect GIMPS layer state information (although this has some implication of violation)
Collect all or selected NSLPs’ state information

On the other hand, the Ping tool could be turned into a stateful tool. A possible function of the Ping tool could then be that it is installing a state on every GIMPS-aware hop it is passing on the way to the Ping message receiver and delete each of the state on the way backwards to the initiator.

4. Summary and Open Issues

We have shown in this document how a testing tool for GIMPS implementations could be designed. Our intentions were to keep it as simple and therefore as portable and extensible as possible. The Ping tool will be able to help GIMPS implementors test their own implementation as well as compare it to others in terms of functionality and basic performance.

Further additions to the Ping tool could be support for tunnelling devices along the GIMPS path and an updated design for a stateful protocol.

5. Security Considerations

A future versions of this document will add security relevant considerations.

6. Acknowledgments

The authors would like to thank Bernd Schloer, Andreas Westermaier and Henning Peters for their feedback.

7. References

7.1 Normative References


7.2 Informative References


Authors’ Addresses

Christian Dickmann  
University of Goettingen  
Telematics Group  
Lotzestr. 16-18  
Goettingen 37083  
Germany  
Email: mail@christian-dickmann.de

Ingo Juchem  
University of Goettingen  
Telematics Group  
Lotzestr. 16-18  
Goettingen 37083  
Germany  
Email: ijuchem@cs.uni-goettingen.de

Sebastian Willert  
University of Goettingen  
Telematics Group  
Lotzestr. 16-18  
Goettingen 37083  
Germany  
Email: swillert@cs.uni-goettingen.de

Xiaoming Fu  
University of Goettingen  
Telematics Group  
Lotzestr. 16-18  
Goettingen 37083  
Germany  
Email: fu@cs.uni-goettingen.de
Intellectual Property Statement

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.

Disclaimer of Validity

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 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.

Copyright Statement

Copyright (C) The Internet Society (2005). 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.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.