H. Schulzrinne
Columbia U.

June 23, 2003

GIMPS: General Internet Messaging Protocol for Signaling
draft-schulzrinne-nsis-ntlp-00

Status of this Memo

Copyright Notice

Abstract

The Generic Internet Messaging Protocol for Signaling (GIMPS) provides a generic transport messaging service to set up, modify and tear down signaling state in signaling nodes.

Table of Contents

1. Requirements notation

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 [1].

2. Introduction

Alternate name: GIST: Generic Internet Signaling Transport

Application-neutral: GIMPS is designed to support the largest range of signaling applications. While a number of such applications have been identified, it appears likely that new applications will emerge. (This was the case after the development of RSVP, for example.)

Mobility support: End systems can change their network attachment point and network address during a session.

Efficient: Signaling often occurs before an application such as an IP telephone conversation can commence, so that any signaling delay becomes noticeable to the application. Signaling delays are incurred by the delay in finding signaling nodes along the path (peer discovery), in retransmitting lost signaling messages and in setting up security associations between nodes, among other factors.

IP version neutral: GIMPS supports both IPv4 and IPv6.

Transport neutral: GIMPS can operate over any message or stream-oriented transport layer, including UDP, DCCP, TCP and SCTP. [TBD: support raw IP?] Messages sent over protocols that do not offer a native fragmentation service, such as UDP, are strictly limited in size and rate to avoid network congestion and loss-amplification problems. [TBD: The 'transport' terminology tends to confuse readers. Maybe we should rename the NTLP as a messaging layer; this document uses the term messaging instead.]

Proxy support: The end systems in a session may not be capable of handling either the signaling transport or the application and may instead rely on proxies to initiate and terminate signaling sessions.

Signaling involves the setting up, modifying and tearing down of state in network elements. GIMPS maintains state along the data path of a data session [ref]. Examples of such state include network resource allotments (for "resource reservation"), a firewall configuration and active network state. Each of these applications is considered a signaling service that uses the transport service defined in this document. Different applications may make use of different services provided by GTSP.

Signaling establishes state sessions, which have a defined beginning and end. While the beginning of a session is always established by

explicit protocol action, a session may end by a signaling teardown message or a time-out ("soft state").

Not every router along the datapath needs to be involved in the signaling session. Indeed, it appears likely that only a subset of nodes will be aware of any given signaling application.

A related set of applications visits nodes along the data path, to discover path properties, for example, but does not leave any state behind. This can be considered a signaling application that establishes and tears down state in the same message and thus is within the scope of this effort.

GIMPS is not an end-to-end transport mechanism for a higher-layer signaling. An example of the latter would be SCTP, used to transport ISUP (SS7 signaling) messages between two nodes. In GIMPS, there are almost always more than two participants in a signaling session, as there is not much point in using a signaling protocol just to communicate between two end points.

GIMPS is not meant to manage application-layer state, but rather to manage state related to data transport. Thus, GIMPS messages need to follow the path of the data. In that crucial respect, it differs from application signaling protocols such as the control component of ftp, SIP [4] and RTSP.

A more detailed discussion can be found in the Next Steps in Signaling Framework [6].

3. Objectives

The signaling transport mechanism has to accomplish two fundamental objectives:

1. Discover the set of nodes along the path from the data sender to the receiver (peer discovery);

2. Deliver signaling information along this chain of nodes.

In many cases, signaling information needs to be delivered reliably between the signaling initiator and responder. Some applications may implement their own reliability mechanism, but experience with RSVP has shown [3] that relying on soft-state refreshes itself may yield unsatisfactory performance if signaling messages are lost even occasionally.

4. Overview of Operations

GIMPS does not attempt to replicate a full-featured transport protocol such as TCP or SCTP. It does not support congestion control, message fragmentation, flow control, acknowledgment windows and selective acknowledgements (SACK). Thus, its "raw" efficiency in more demanding network conditions is likely to be low. Instead, GIMPS leverages the continuing advances in transport protocols such as TCP and SCTP for messages where these features are useful. For small messages and discovery, it uses UDP [or raw IP.]

Each node maintains a forwarding state table that includes

session identifier: Cryptographically random and globally unique session identifier;

destination address: The destination address of the message, contained in the GIMPS message. (This is not necessarily the IP address in the message.)

Generally, each session will have at least two entries, one for the initiator-to-responder direction, the other for the responder-to-initiator message flow. If the end points are mobile, additional entries may be added. The forwarding state table entries are discarded after the Rediscovery Period (RDP).

For efficiency, GIMPS offers two modes a operation, a "datagram" mode for small, infrequent messages with modest delay constraint and a "connection" mode for larger data objects or where fast setup in the face of packet loss is desirable. The datagram mode can use any lower-layer unreliable datagram transport mechanism, with UDP as the initial choice. The connection mode can use any stream or message-oriented transport protocol, including TCP and SCTP.

On receiving a GIMPS message, a node performs the operations described below. (It does not matter whether the message arrived over a reliable or unreliable lower-layer transport mechanism.)

Below, we call the GIMPS node that tries to determine the next-hop peer the querying node.

1. The GIMPS node compares the GIMPS destination network address (not the lower-layer network address) to its own address. If it matches one of its addresses, the message has arrived and is passed to the signaling application for further processing.

2. The GIMPS node inspects the session identifier in the incoming message and determines if it matches an existing session. It

also compares the responder address to the responder contained in the state record. If both match and the rediscovery period (RDP) has not expired, the node forwards the message to the next node on the existing transport and security association (e.g., TCP connection, TLS session, or IPsec session).

If there is no known next-hop, the node checks the message size and compares it against the maximum datagram size (MDS, below), a global constant. (Since the message may be forwarded across multiple hops, knowledge of the link MTU size is not sufficient.) If the message size falls below MDS, the message is forwarded towards the network address contained in the GIMPS message, i.e., the current responder and marked with an IP router-alert option that causes it to be intercepted by the next GIMPS-capable node. The GIMPS message uses the source address of this node, to facilitate the discovery of network problems and to allow the next node to return a confirmation message (see below).

If the message size exceeds MDS, the node constructs a discovery message that has the same message type, session identifier and client-layer identifier as the GIMPS message triggering it. It then transmits it in the same manner described in the last paragraph.

Messages that arrive during the discovery phase can be queued or also sent forth as discovery messages. Messages that exceed MDS in size MUST be queued. To avoid network congestion, a node MUST NOT have more than one message outstanding at any given time. If no response is received within the retransmission interval (RTI, default 1 s), the message is retransmitted. (No instance identifier is used since round-trip time estimation is unlikely to be successful.)

The node records the transport association or network address of the previous hop. This information is used for messages that are sent by the responder to the initiator.

3. When the next node receives a GIMPS message with the 'response-requested' flag, it sends a response to the IP address of that message, confirming receipt. The response uses the source address of the next-hop node and is addressed to the querying node. The response includes a cookie that is used to prevent denial-of-service (state-exhaustion) attacks by nodes spoofing the source address in the GIMPS message. The node only establishes the GIMPS session if it contains a valid cookie.

4. When a node receives a message, regardless of the transport protocol, the node records the transport association that the

message arrived on in the state table. This information is then used to route messages in the opposite direction. For example, if a discovery message arrived with a source address of A and a destination address of B, the node records that any message with destination address B can reach B via that association.

5. When a node receives a response to a pending discovery message, it determines if there is an existing transport and/or security association with that node. If not, it establishes such a connection or association. (The response indicates the types of security and transport mechanisms that are available, e.g., TLS-over-SCTP, UDP, etc.)

   In either case, the GIMPS node sends any queued messages on that new or existing association. If the message indicates the error condition that no state was established, the node extracts the cookie from the message and tries again, this time addressing the message to the correct next-hop destination.

5. Transport Usage

As noted above, GIMPS can operate in a datagram mode, for peer discovery and short-message delivery, and in connection mode, for messages that exceed the size threshold MDS (typically, 500 bytes). Nodes MUST support both modes, but applications can be structured so that they only use one or the other mode. Connection mode requires the datagram mode for data-path peer discovery; in the future, there may be other peer-discovery mechanism that do not require sending data. However, these are beyond the scope of this document.

It is possible to combine these two modes along a chain of nodes, without coordinationor manual configuration. This allows, for example, the use of datagram modes at the edges of the network and connection-oriented operation in the core of the network. Such combinations may make operation more efficient for mobile endpoints, while allowing multiplexing of signaling messages across shared security and transport associations between core routers.

6. Message Format

The following items are contained in each GIMPS message:

Initiator address: The current network (IPv4 or IPv6) address of the initiator of the signaling session. The initiator may change during a session, e.g., if the initiator moves to a different network.

Responder address: The current network (IPv4 or IPv6) address of the destination (responder) of the signaling session. The responder may change during a session, e.g., if the initiator moves to a different network.

Session identifier: The GIMPS session identifier is a long, cryptographically random identifier chosen by the initiator. The length is TBD, but 128 bits should be more than sufficient to make the probability of collisions orders of magnitude lower than other failure reasons.

Hop counter: A hop counter prevents a message from looping indefinitely. (Since messages may get translated between different lower-layer transport protocols, the IP hop count cannot be relied upon.)

Service identifier: The service identifier [TBD: application identifier?] describes the signaling application, such as resource reservation or firewall control.

Message identifier: A four-octet message counter, used to associate messages with their confirmations.

Cookies: Each message contains two X-octet cookies, generated for each hop. The cookie in the next request with the same session identifier and needs to be designed so that a node can determine the validity of a cookie without keeping state.

Flags: A number of flags define protocol operations, such as "confirmation requested" (hop-by-hop confirmation message).

Message type: The operation code defines three operations:

establish: Establish or refresh a session.

refresh: Refresh only if the session exists [TBD: is this useful?]

failure: A message-layer failure occurred, such as a mis-formatted message or an authentication or integrity check failure.

teardown: Tear down.

confirmation: Confirms the receipt of an earlier message, with the message number included.

The following items are optional:

Lifetime: The lifetime of a session in the absence of refreshes, measured in seconds. Defaults to 30 seconds. Cannot be changed by any intermediate node.

Confirm: Confirms receipt of a message. [May not be needed if 'confirmation' automatically means that the message number is confirmed.]

The message content is encoded in an RSVP-style format, i.e., consisting of type-length-value (TLV) objects. If transported on a bytestream-oriented protocol, the whole message is preceded by a four-octet length field.

7. Security Considerations

7.1 Confidentiality

GIMPS can use lower-layer transport functionality, such as TLS or IPsec, to ensure message confidentiality. In many cases, confidentiality of messages is not likely to be a prime concern at the messaging layer, in particular since messages are often sent to parties which are unknown ahead of time. Signaling applications will likely have their own mechanism for securing content as necessary.

7.2 Integrity

GIMPS can use lower-layer hop-by-hop transport functionality, such as TLS or IPsec, to ensure message integrity. Message-layer cryptographic integrity protection requires a shared secret or that the receiver knows the public key of the sender. Some components of the message, such as the hop count, will need to be modified by GIMPS nodes, so that only hop-by-hop integrity is likely to be useful for the messaging layer part. The use of CMS [5] encapsulation is RECOMMENDED.

7.3 Authentication

GIMPS nodes can assure themselves of the identity of the next hop via the the lower-layer transport functionality. However, with discovery, there is no effective way to know what is the legitimate next hop as opposed to an impostor.

7.4 Denial of Service Prevention

GIMPS is designed so that each connection-less discovery message only generates at most one response, so that a GIMPS node cannot become the source of a denial of service attack.

However, GIMPS can still be subjected to denial-of-service attacks where an attacker using forged source addresses forces a note to establish state without return routability, causing a problem similar to TCP SYN flood attacks. There are two types of state attacks and one computational resource attack. In the first state attack, an attacker floods a node with messages that the node has to store until it can determine the next hop. If the destination address is chosen so that there is no next hop, the node would accumulate messages for several seconds until the discovery retransmission attempt times out. The second type of state-based attack causes GIMPS state to be established by bogus messages. A related computational/ network-resource attack uses unverified messages to cause a node to make AAA queries or attempt to cryptographically verify a digital

signature. (RSVP is vulnerable to this type of attack.)

There are at least three defenses against these attacks:

1. The receiving node does not establish a session or discover its next hop on receiving the unreliable (discovery) message, but rather waits for a setup message on a reliable channel. If the reliable channel exists, the additional delay is one one-way delay and is no more than the minimal theoretically possible delay of a three-way handshake, i.e., 1.5 node-to-node round-trip times. The delay gets significantly larger if a new connection needs to be established first.

2. The response to the initial discovery message contains a cookie. The previous hop repeats the discovery with the cookie included. State is only established for messages that contain a valid cookie. The setup delay is also 1.5 round-trip times. (This mechanism is similar to that in SCTP [2].)

3. If there is a chance that the next-hop nodes shares a secret with the previous hop, the sender could include a hash of the session ID and the sender's secret. The receiver can then verify that the message was likely sent by the purported source. This does not scale well, but may work if most nodes tend to communicate with a small peer clique of nodes. (In that case, however, they might as well establish more-or-less permanent transport sessions with each other.)

These techniques are complementary; we chose a combination of the first and second method.

Normative References

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

[2] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[3] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F. and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

[4] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[5] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.

[6] Hancock, R., "Next Steps in Signaling: Framework", draft-ietf-nsis-fw-02 (work in progress), March 2003.

Author's Address

Henning Schulzrinne
Columbia University
Department of Computer Science
450 Computer Science Building
New York, NY 10027
US

Phone: +1 212 939 7042
EMail: hgs+nsis@cs.columbia.edu
URI: http://www.cs.columbia.edu

Appendix A. Acknowledgements

This document is based on the discussions within the IETF NSIS working group. The comments by ... helped improve the document.

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

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