Next Steps in Signaling
Internet-Draft
Expires: November 28, 2004

H. Schulzrinne
Columbia U.
R. Hancock
Siemens/RMR
May 30, 2004

GIMPS: General Internet Messaging Protocol for Signaling
draft-ietf-nsis-ntlp-02

Status of this Memo

Copyright Notice

Abstract

This document specifies protocol stacks for the routing and transport of per-flow signaling messages along the path taken by that flow through the network. The design uses existing transport and security protocols under a common messaging layer, the General Internet Messaging Protocol for Signaling (GIMPS), which provides a universal service for diverse signaling applications. GIMPS does not handle signaling application state itself, but manages its own internal state and the configuration of the underlying transport and security

protocols to enable the transfer of messages in both directions along the flow path. The combination of GIMPS and the lower layer protocols provides a solution for the base protocol component of the "Next Steps in Signaling" framework.

Table of Contents

1. Introduction

Signaling involves the manipulation of state held in network elements. 'Manipulation' could mean setting up, modifying and tearing down state; or it could simply mean the monitoring of state which is managed by other mechanisms.

This specification concentrates specifically on the case of "path-coupled" signaling, which involves network elements which are located on the path taken by a particular data flow, possibly including but not limited to the flow endpoints. Indeed, there are almost always more than two participants in a path-coupled-signaling session, although there is no need for every router on the path to participate. Path-coupled signaling thus excludes end-to-end higher-layer application signaling (except as a degenerate case) such as ISUP (telephony signaling for Signaling System #7) messages being transported by SCTP between two nodes.

In the context of path-coupled signaling, examples of state management include network resource allocation (for "resource reservation"), firewall configuration, and state used in active networking; examples of state monitoring are the discovery of instantaneous path properties (such as available bandwidth, or cumulative queuing delay). Each of these different uses of path-coupled signaling is referred to as a signaling application.

Every signaling application requires a set of state management rules, as well as protocol support to exchange messages along the data path. Several aspects of this support are common to all or a large number of applications, and hence should be developed as a common protocol. The framework given in [22] provides a rationale for a function split between the common and application specific protocols, and gives outline requirements for the former, the 'NSIS Transport Layer Protocol' (NTLP).

This specification provides a concrete solution for the NTLP. It is based on the use of existing transport and security protocols under a common messaging layer, the General Internet Messaging Protocol for Signaling (GIMPS). Different signaling applications may make use of different services provided by GIMPS, but GIMPS does not handle signaling application state itself; in that crucial respect, it differs from application signaling protocols such as the control component of FTP, SIP and RTSP. Instead, GIMPS manages its own internal state and the configuration of the underlying transport and security protocols to ensure the transfer of signaling messages on behalf of signaling applications in both directions along the flow path.

1.1 Restrictions on Scope

This section briefly lists some important restrictions on GIMPS applicability and functionality. In some cases, these are implicit consequences of the functionality splits developed in the framework; in others, they are restrictions on the types of scenario in which GIMPS can operate correctly.

Flow splitting: In some cases, e.g. where packet-level load sharing has been implemented, the path taken by a single flow in the network may not be well defined. If this is the case, GIMPS cannot route signaling meaningfully. (In some circumstances, GIMPS can detect this condition, but this cannot be guaranteed.)

Multicast: GIMPS does not handle multicast flows. This includes 'classical' IP multicast and any of the 'small group multicast' schemes recently proposed.

2. Requirements Notation and Terminology

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 [2].

The terminology used in this specification is fully defined in this section. The basic entities relevant at the GIMPS level are shown in Figure 1.

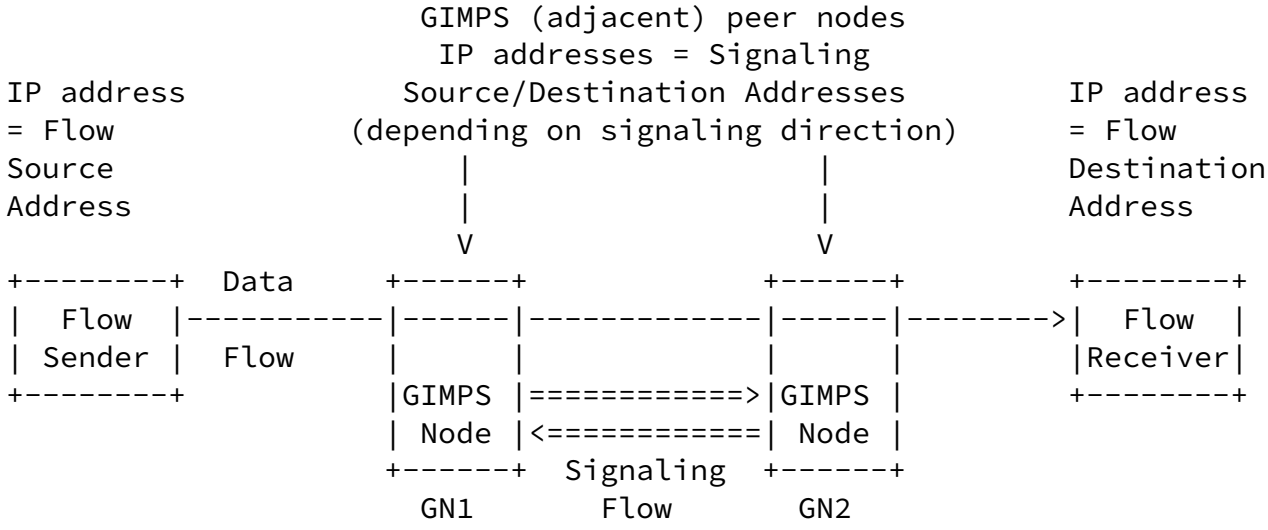


Figure 1: Basic Terminology

[Data] Flow: A set of packets identified by some fixed combination of header fields. Flows are unidirectional (a bidirectional communication is considered a pair of unidirectional flows).

Session: A single application layer flow of information for which some network control state information is to be manipulated or monitored. IP mobility may cause the mapping between sessions and flows to change, and IP multihoming may mean there is more than one flow for a given session.

[Flow] Sender: The node in the network which is the source of the packets in a flow. Could be a host or a router (if the flow is actually an aggregate).

[Flow] Receiver: The node in the network which is the sink for the packets in a flow.

Downstream: In the same direction as the data flow.

Upstream: In the opposite direction to the data flow.

GIMPS Node: Any node along the data path supporting GIMPS (regardless of what signaling applications it supports).

Adjacent peer: The next GIMPS node along the data path, in the upstream or downstream direction. Whether two nodes are adjacent is determined implicitly by the GIMPS peer discovery mechanisms; it is possible for adjacencies to 'skip over' intermediate GIMPS nodes if they have no interest in the signaling messages being exchanged.

Datagram mode: A mode of sending GIMPS messages between nodes without using any transport layer state or security protection. Upstream messages are sent UDP encapsulated directly to the signaling destination; downstream messages are sent towards the flow receiver with a router alert option.

Connection mode: A mode of sending GIMPS messages directly between nodes using point to point "messaging associations" (see below), i.e. transport protocols and security associations.

Messaging association: A single connection between two explicitly identified GIMPS adjacent peers, i.e. between a given signaling source and destination address. A messaging association uses a specific transport protocol and known ports, and may be run over specific network layer security associations, or use a transport layer security association internally. A messaging association is bidirectional.

3. Design Methodology

3.1 Overall Approach

The generic requirements identified in [22] for transport of path-coupled signaling messages are essentially two-fold:

"Routing": Determine how to reach the adjacent signaling node along the data path (the GIMPS peer);

"Transport": Deliver the signaling information to that peer.

To meet the routing requirement, for downstream signaling the node can either use local state information (e.g. gathered during previous signaling exchanges) to determine the identity of the GIMPS peer explicitly, or it can just send the signaling towards the flow destination address and rely on the peer to intercept it. For upstream signaling, only the first technique is possible.

Once the routing decision has been made, the node has to select a mechanism for transport of the message to the peer. GIMPS divides the transport problems into two categories, the easy and the difficult ones. It handles the easy cases within GIMPS itself, avoiding complexity and latency, while drawing on the services of well-understood reliable transport protocols for the harder cases. Here, with details discussed later, "easy" messages are those that are sized well below the lowest MTU along a path, are infrequent enough not to cause concerns about congestion and flow control, and do not need transport or network-layer security protection.

However, in many cases, signaling information needs to be delivered between GIMPS peers with additional transport or security properties. For example, signaling applications could implement their own reliability mechanism, but experience with RSVP has shown [14] that relying solely on soft-state refreshes may yield unsatisfactory performance if signaling messages are lost even occasionally. The provision of this type of reliability is therefore also the responsibility of the underlying transport protocols.

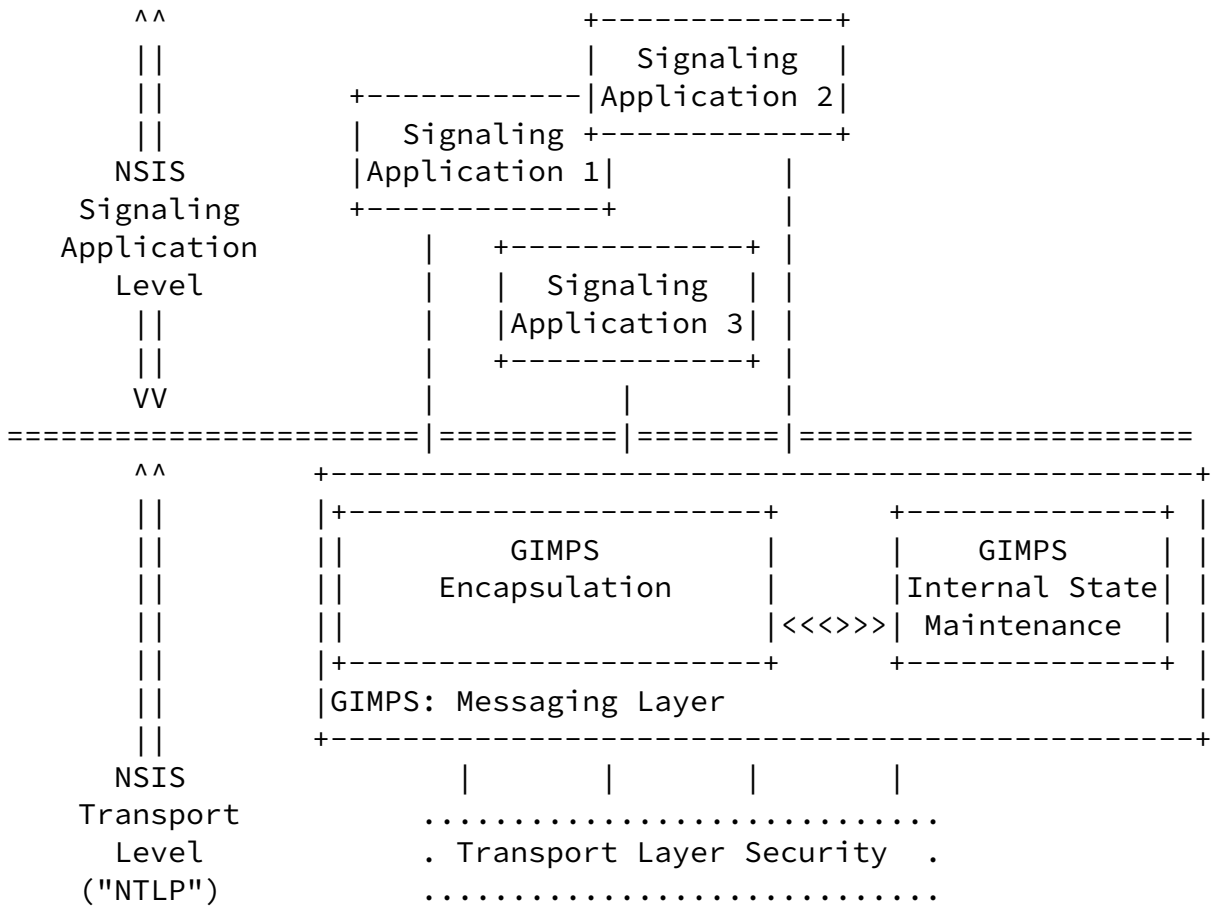
In [22] all of these routing and transport requirements are assigned to a single notional protocol, the 'NSIS Transport Layer Protocol' (NTLP). The strategy of splitting the transport problem leads to a layered structure for the NTLP, as a specialised GIMPS 'messaging' layer running over standard transport and security protocols, as

shown in Figure 2.

GIMPS offers two modes of transport operation:

Datagram mode: for small, infrequent messages with modest delay constraints; and

Connection mode: for larger data objects or where fast setup in the face of packet loss is desirable, or where channel security is required.



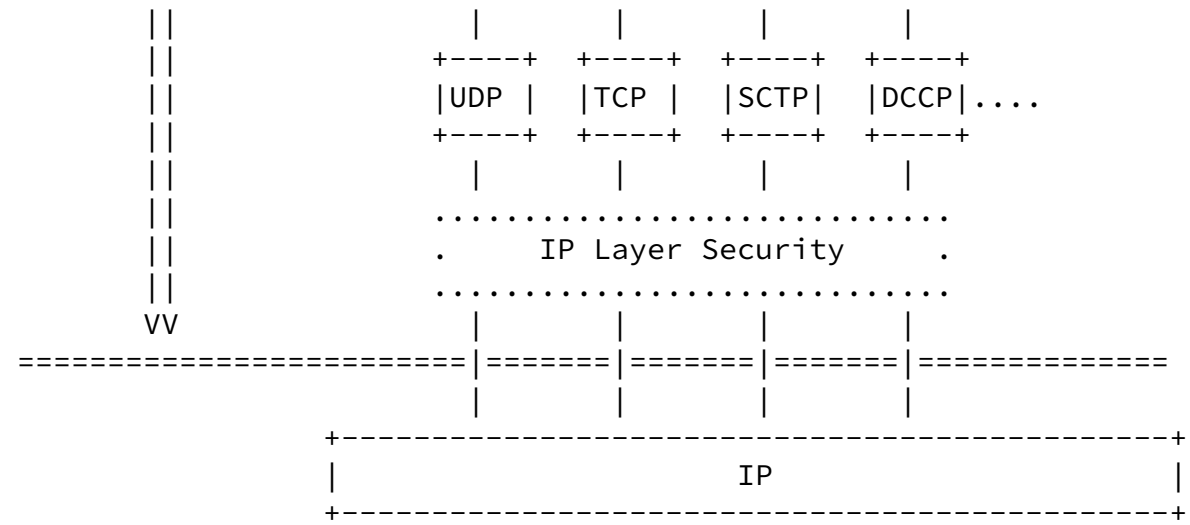


Figure 2: Protocol Stacks for Signaling Transport

The datagram mode uses an unreliable unsecured datagram transport mechanism, with UDP as the initial choice. The connection mode can use any stream or message-oriented transport protocol. It may employ specific network layer security associations (e.g. IPsec), or an internal transport layer security association (e.g. TLS).

It is possible to mix these two modes along a chain of nodes, without coordination or manual configuration. This allows, for example, the use of datagram mode at the edges of the network and connection mode 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 associations and transport connections between core routers.

It must be understood that the routing and transport decisions made by GIMPS are not totally independent. If the message transfer has requirements that enforce the use of connection mode (e.g. because fragmentation is required), this can only be used between explicitly identified nodes. In such cases, the GIMPS node must carry out signaling in datagram mode to identify the peer and then set up the

necessary transport connection. The datagram mode option of sending the message in the direction of the flow receiver and relying on interception is not available.

In general, the state associated with connection mode messaging to a particular peer (signaling destination address, protocol and port numbers, internal protocol configuration and state information) is referred to as a "messaging association". There may be any number of messaging associations between two GIMPS peers (although the usual case is 0 or 1), and they are set up and torn down by management actions within GIMPS itself.

3.2 Design Attributes

Soft state: All parts of GIMPS state are subject to time-out ("soft-state"). 'State' here includes the messaging associations managed by GIMPS.

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 ones will emerge.

Mobility support: End systems can change their network attachment point and network address during a session. GIMPS minimises the use of IP addresses as identifiers for non-topological information (e.g. authentication state).

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. GIMPS attempts to minimise these delays by several mechanisms, such as the use of high performance transport protocols to circumvent message loss, and the re-use of messaging associations to avoid setup latency. If explicit discovery is needed it is a lightweight process which only probes local topology, and GIMPS also allows it to be bypassed completely for downstream datagram mode messages.

IP version neutral: GIMPS supports both IPv4 and IPv6: it can use either for transport, largely as a result of their support in the underlying transport protocols, and can signal for either type of flow. In addition, GIMPS is able to operate on dual-stack nodes (to bridge between v4 and v6 regions) and also to operate across v4/v6 boundaries and other addressing boundaries. Specific transition issues are considered in Section 6.5.

Transport neutral: GIMPS can operate over any message or stream-oriented transport layer, including UDP, DCCP, TCP and SCTP. Messages sent over protocols that do not offer a native fragmentation service, such as UDP or DCCP, are strictly limited in size to avoid loss-amplification; in the case of UDP, they must also be limited in rate to avoid network congestion.

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. Proxy support is limited to nodes that are actually on the data path (for example, access routers for stub networks); signaling from a 3rd party node not associated with the data path is not considered. GIMPS decouples the operation of the messaging functions from the flow source and destination addresses, treating these primarily as data.

Scaleable: As will be discussed in Section 4.3, up to one messaging association is generally kept for each adjacent GIMPS peer and thus association state scales better than the number of sessions. (Many peers may not have association state at all, if there are no messages for sessions visiting those nodes that warrant such treatment.) Messaging associations are managed based on policy at each node, depending on trade-offs between fast peer-to-peer communication and state overhead. Messaging association state can be removed immediately after the last signaling session to a

particular next-hop is removed, after some delay to wait for new sessions, or only if resource demands warrant it.

3.3 Example of Operation

This section presents an example of GIMPS usage in a relatively simple (in particular, NAT-free) signaling scenario, to illustrate its main features.

Consider the case of an RSVP-like signaling application which allocates resources for a flow from sender to receiver; we will consider how GIMPS transfers messages between two adjacent peers along the path, GN1 and GN2 (see Figure 1). In this example, the end-to-end exchange is initiated by the signaling application instance in the sender; we take up the story at the point where the first message is being processed (above the GIMPS layer) by the signaling application in GN1.

1. The signaling application in GN1 determines that this message is a simple description of resources that would be appropriate for the flow. It determines that it has no special security or transport requirements for the message, but simply that it should be transferred to the next downstream signaling application peer on the path that the flow will take.

2. The message payload is passed to the GIMPS layer in GN1, along with a definition of the flow and description of the transfer requirements {downstream, unsecured, unreliable}. GIMPS determines that this particular message does not require fragmentation and that it has no knowledge of the next peer for this flow and signaling application; however, it also determines that this application is likely to require secured upstream and downstream transport of large messages in the future. This determination is a function of node-local policy, and some options for how it may be communicated between NSLP and GIMPS implementations within a node are indicated in Appendix D.

3. GN1 therefore constructs a UDP datagram with the signaling

application payload, and additional payloads at the GIMPS level to be used to initiate the possible setup of a messaging association. This datagram is injected into the network, addressed towards the flow destination and with a Router Alert Option included.

4. This D-mode message passes through the network towards the flow receiver, and is seen by each router in turn. GIMPS-unaware routers will not recognise the RAO value and will forward the

message unchanged; GIMPS-aware routers which do not support the signaling application in question will also forward the message unchanged, although they may need to process more of the message to decide this.

5. The message is finally intercepted at GN2. The GIMPS layer identifies that the message is relevant to a local signaling application, and passes the signaling application payload and flow description to upwards to it. From there, the signaling application in GN2 can continue to process this message as in GN1 (compare step 1), and this will eventually result in the message reaching the flow receiver.

6. In parallel, the GIMPS instance in GN2 recognises that GN1 is attempting to discover GN2 in order to set up a messaging association for future signaling for the flow. There are two possible cases:

   A. GN1 and GN2 already have an appropriate association. GN2 simply records the identity of GN1 as its upstream peer for that flow and signaling application, and sends a GIMPS message back to GN1 over the association identifying itself as the peer for this flow.

   B. No messaging association exists. Again, GN2 records the

identity of GN1 as before, but sends an upstream D-mode message to GN1, identifying itself and agreeing to the association setup. The protocol exchanges needed to complete this will proceed in the background, controlled by GN1.

7. Eventually, another signaling application message works its way upstream from the receiver to GN2. This message contains a description of the actual resources requested, along with authorisation and other security information. The signaling application in GN2 passes this payload to the GIMPS level, along with the flow definition and transfer requirements {upstream, secured, reliable}.

8. The GIMPS layer in GN2 identifies the upstream peer for this flow and signaling application as GN1, and determines that it has a messaging association with the appropriate properties. The message is queued on the association for transmission (this may mean some delay if the negotiations begun in step 6.B have not yet completed).

Further messages can be passed in each direction in the same way. The GIMPS layer in each node can in parallel carry out maintenance operations such as route change detection (this can be done by

sending additional GIMPS-only datagram mode messages, see Section 6.1 for more details).

It should be understood that many of these details of GIMPS operations can be varied, either by local policy or according to signaling application requirements, and they are also subject to development and refinement as the protocol design proceeds. The authoritative details are contained in the remainder of this document.

4. GIMPS Processing Overview

This section defines the basic structure and operation of GIMPS. It is divided into three parts. Section 4.1 gives an overview of the per-flow and per-peer state that GIMPS maintains for the purpose of transferring messages. Section 4.2 describes how messages are processed in the case where any necessary messaging associations and associated routing state already exist; this includes the simple scenario of pure datagram mode operation, where no messaging associations are necessary in the first place (equivalent to the transport functionality of base RSVP as defined in [9]). Section 4.3 describes how routing state is maintained and how messaging associations are initiated and terminated.

4.1 GIMPS State

4.1.1 Message Routing State

For each flow, the GIMPS layer can maintain message routing state to manage the processing of outgoing messages. This state is conceptually organised into a table with the following structure.

The primary key (index) for the table is the combination of the information about how the message is to be routed, the session being signalled for, and the signaling application itself:

Message Routing Information (MRI): This defines the method to be used to route the message, and any associated addressing information. In the simplest case, the message routing method is to follow the path that is being taken by the data flow, and the associated addressing is the flow header N-tuple (i.e. the Flow-Identifier of [22]).

Session Identification (SID): This is a cryptographically random and (probabilistically) globally unique identifier of the application layer session that is using the flow. For a given flow, different signaling applications may or may not use the same session identifier. Often there will only be one flow for a given session, but in mobility/multihoming scenarios there may be more than one and they may be differently routed.

Signaling Application Identification (NSLPID): This is an IANA assigned identifier of the signaling application which is generating messages for this flow. The inclusion of this identifier allows the routing state to be different for different signaling applications (e.g. because of different adjacencies).

The state information for a given key is as follows:

Upstream peer: the adjacent GIMPS peer closer to the flow source. This could be an IP address (learned from previous signaling) or a pointer to a messaging association. It could also be null, if this node is the flow sender or this node is not storing reverse routing state, or a special value to indicate that this node is the last upstream node (but not the sender).

Downstream peer: the adjacent GIMPS peer closer to the flow destination. This could be a pointer to a messaging association, or it could be null, if this node is the flow receiver or this node is only sending downstream datagram mode messages for this flow and signaling application, or a special value to indicate that this node is the last downstream node (but not the receiver).

Note that both the upstream and downstream peer state may be null, and that the session identifier information is not actually required for message processing; in that case, no state information at all needs to be stored in the table. Both items of peer identification state have associated timers for how long the identification can be considered accurate; when these timers expire, the peer identification (IP address or messaging association pointer) is purged if it has not been refreshed. An example of a routing state table for a simple scenario is given in Appendix B.

Note also that the information is described as a table of flows, but that there is no implied constraint on how the information is stored. For example, in a network using pure destination address routing (without load sharing or any form of policy-based forwarding), the downstream peer information might be possible to store in an aggregated form in the same manner as the IP forwarding table. In addition, many of the per-flow entries may point to the same per-peer state (e.g. the same messaging association) if the flows go through the same adjacent peer. However, in general, and especially if GIMPS

peers are several IP hops away, there is no way to identify the correct downstream peer for a flow and signaling application from the local forwarding table using prefix matching, and the same applies always to upstream peer state because of the possibility of asymmetric routing. Per-flow routing state has to be stored, just as for RSVP [9].

4.1.2 Messaging Association State

The per-flow message routing state is not the only state stored by GIMPS. There is also the state required to manage the messaging associations. Since we assume that these associations are typically per-peer rather than per-flow, they are stored in a separate table, including the following information:

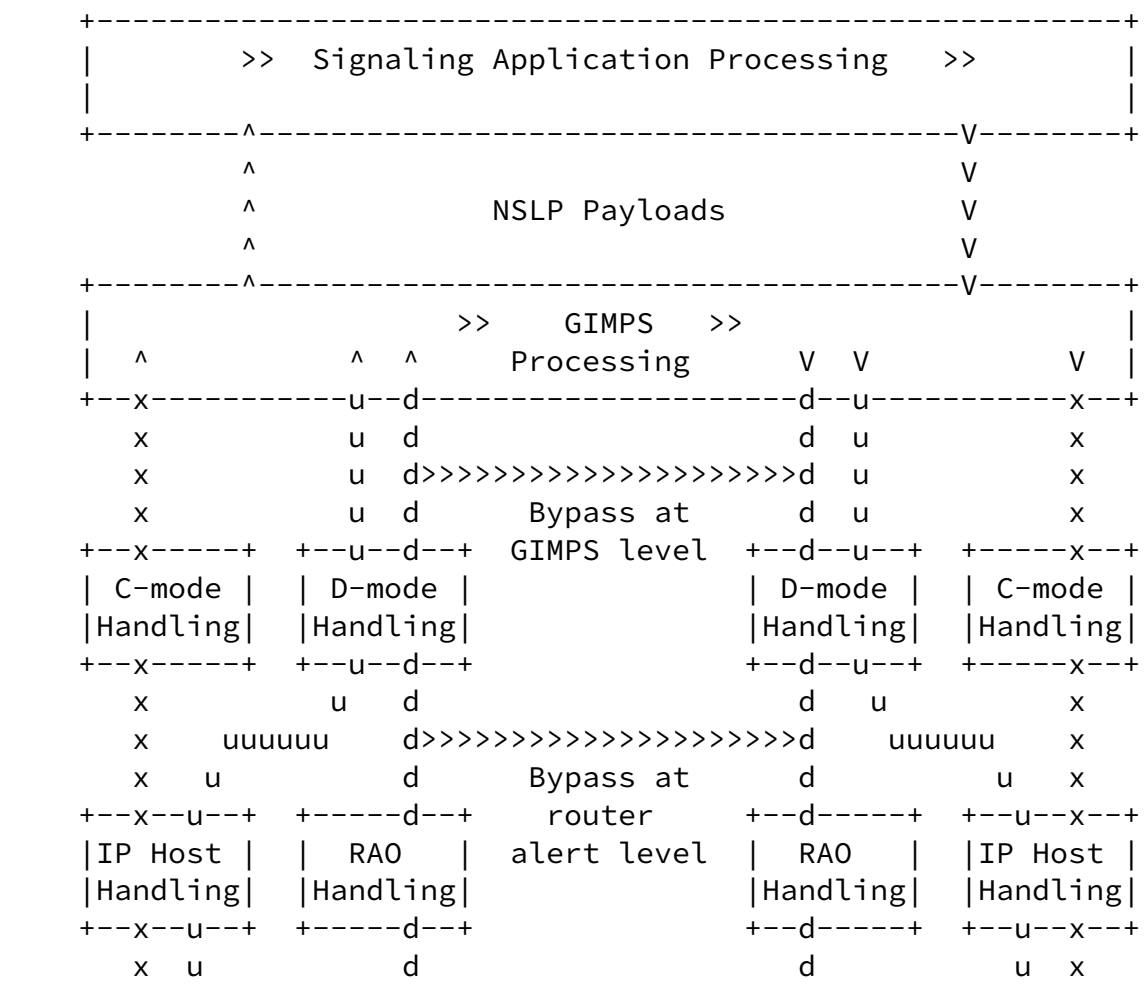
- messages pending transmission while an association is being established;
- an inactivity timer for how long the association has been idle.

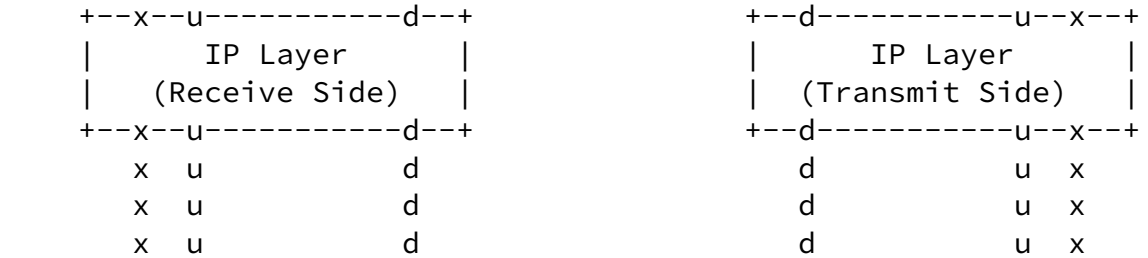
In addition, per-association state is held in the messaging association protocols themselves. However, the details of this state are not directly visible to GIMPS, and they do not affect the rest of the protocol description.

4.2 Basic Message Processing

This section describes how signaling application messages are processed in the simple case where any necessary messaging associations and routing state are already in place. The description is divided into several parts. Firstly, message reception, local processing and message transmission are described for the case where the node handles the NSLPID in the message. Secondly, the case where

the message is forwarded directly in the IP or GIMPS layer (because there is no matching signaling application on the node) is given. An overview is given in Figure 3.





```
uuuuuuuuuuuuuu = upstream datagram mode messages
dddddddddddddd = downstream datagram mode messages
xxxxxxxxxxxxxx = connection mode messages
           RAO = Router Alert Option
```

Figure 3: Message Paths through a GIMPS Node

Note that the same messages are used for maintaining internal GIMPS state and carrying signaling application payloads. The state maintenance takes place as a result of processing specific GIMPS payloads in these messages. The processing of these payloads is the subject of Section 4.3.

4.2.1 Message Reception

Messages can be received in connection or datagram mode, and from upstream or downstream peers.

Reception in connection mode is simple: incoming packets undergo the security and transport treatment associated with the messaging association, and the messaging association provides complete messages to the GIMPS layer for further processing. Unless the message is protected by a query/response cookie exchange (see Section 4.3, the routing state table is checked to ensure that this messaging association is associated with the MRI/SID/NSLPID combination.

Reception in datagram mode depends on the message direction. Upstream messages (from a downstream peer) will arrive UDP encapsulated and addressed directly to the receiving signaling node. Each datagram contains a single complete message which is passed to the GIMPS layer for further processing, just as in the connection mode case.

Downstream datagram mode messages are UDP encapsulated with an IP router alert option to cause interception. The signaling node will therefore 'see' all such messages. The case where the NSLPID does not match a local signaling application is considered below in Section 4.2.4; otherwise, it is passed up to the GIMPS layer for further processing as in the other cases.

4.2.2 Local Processing

Once a message has been received, by any method, it is processed locally within the GIMPS layer. The GIMPS processing to be done depends on the payloads carried; most of the GIMPS-internal payloads are associated with state maintenance and are covered in Section 4.3.

One GIMPS-internal payload which is carried in each message and requires processing is the GIMPS hop count. This is decremented on input processing, and checked to be greater than zero on output processing. The primary purpose of the GIMPS hop count is to prevent message looping.

The remainder of the GIMPS message consists of an NSLP payload. This is delivered locally to the signaling application identified at the GIMPS level; the format of the NSLP payload is not constrained by GIMPS, and the content is not interpreted.

Signaling applications can generate their messages for transmission, either asynchronously, or in response to an input message, and GIMPS can also generate messages autonomously. Regardless of the source, outgoing messages are passed downwards for message transmission.

4.2.3 Message Transmission

When a message is available for transmission, GIMPS uses internal policy and the stored routing state to determine how to handle it. The following processing applies equally to locally generated messages and messages forwarded from within the GIMPS or signaling application levels.

The main decision is whether the message must be sent in connection mode or datagram mode. Reasons for using the former could be:

- NSLP requirements: for example, the signaling application has requested channel secured delivery, or reliable delivery;

- protocol specification: for example, this document could specify that a message that requires fragmentation MUST be sent over a messaging association;

- local GIMPS policy: for example, a node may prefer to send messages over a messaging association to benefit from congestion control.

In principle, as well as determining that some messaging association must be used, GIMPS could select between a set of alternatives, e.g. for load sharing or because different messaging associations provide

different transport or security attributes (see Section 8.5 for further discussion).

If the use of a messaging association is selected, the message is queued on the association (found from the upstream or downstream peer state table), and further output processing is carried out according to the details of the protocol stack used for the association. If no appropriate association exists, the message is queued while one is created (see Section 4.3). If no association can be created, this is again an error condition, and should be indicated back to the NSLP.

If a messaging association is not required, the message is sent in datagram mode. The processing in this case depends on whether the message is directed upstream or downstream.

- If the upstream peer IP address is available from the per-flow routing table, the message is UDP encapsulated and sent directly to that address. Otherwise, the message cannot be forwarded (i.e. this is again an error condition).

- o In the downstream direction, messages can always be sent. They are simply UDP encapsulated and IP addressed using information from the MRI, with the appropriate router alert option.

4.2.4 Bypass Forwarding

A GIMPS node may have to handle messages for which it has no signaling application corresponding to the message NSLPID. There are several possible cases depending mainly on the RAO setting (see Section 8.4 for more details):

A downstream datagram mode message contains an RAO value associated with NSIS, and the IP layer is unable to determine whether to forward it.

A downstream datagram mode message contains an RAO value which is relevant to the node, but the signaling application for the actual NSLPID is not processed.

A message is delivered directly (e.g. in C-mode) to the node for which there is no corresponding signaling application. (According to the rules of the current specification, this should never happen. However, future versions might find a use for such a feature.)

In all cases, the role of GIMPS is to forward the message essentially unchanged. However, a GIMPS implementation must ensure that the IP TTL field and GIMPS hop count are managed correctly to prevent

message looping, and this should be done consistently independently of whether the processing (e.g. for case (1)) takes place on the fast path or in GIMPS-specific code. The rules are that in cases (1)

and (2), the IP TTL is decremented just as if the message was a normal IP forwarded packet; in cases (2) and (3) the GIMPS hop count is decremented as in the case of normal input processing.

4.3 Routing State and Messaging Association Maintenance

The main responsibility of the GIMPS layer is to manage the routing state and messaging associations which are used in the basic message processing described above. Routing state is installed and maintained by datagram mode messages containing specific GIMPS payloads. Messaging associations are dependent on the existence of routing state, but are actually set up by the normal procedures of the transport and security protocols that comprise the messaging association. Timers control routing state and messaging association refresh and expiration.

The complete sequence of possible messages for state setup between adjacent peers is shown in Figure 4 and described in detail in the following text.

The initial message in any routing state maintenance operation is a downstream datagram mode message, sent from the querying node and intercepted at the responding node. This is encapsulated and addressed just as in the normal case; in particular, it has addressing and other identifiers appropriate for the flow and signaling application that state maintenance is being done for, and it is allowed to contain an NSLP payload. Processing at the querying and responding nodes is also essentially the same. However, the querying node includes additional payloads: its own address information, a proposal for possible messaging association protocol stacks, and optionally 'Discover-Query' information, including a Response Request flag and a Query Cookie. This message is informally referred to as a 'GIMPS-query'. The role of the cookies in this and subsequent messages is to protect against certain denial of service attacks.

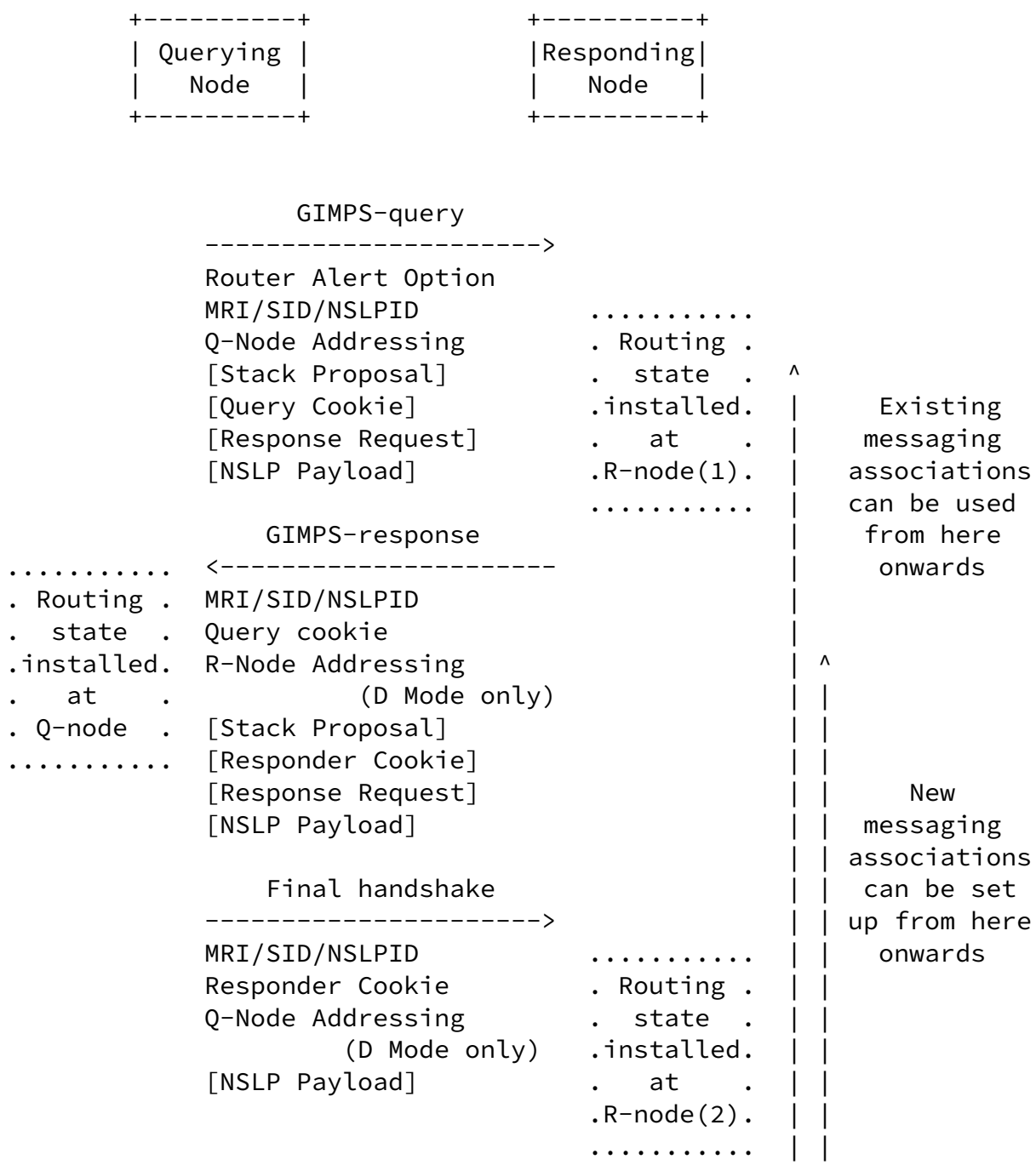


Figure 4: Message Sequence at State Setup

In the responding node, the GIMPS level processing of the Discover-Query information triggers the generation of a 'GIMPS-response' message. This is also a normally encapsulated and addressed message with particular payloads, this time in the upstream direction. Again, it can contain an NSLP payload (possibly a response to the NSLP payload in the initial message). It includes its own addressing information, a counter-proposal for messaging association protocol stacks, and the Query Cookie, and optionally 'Discover-Response' information, including another Response Request flag and a Responder Cookie. Note that if a messaging association

already exists towards the querying node, this can be used to deliver the GIMPS-response message; otherwise, datagram mode is used.

The querying node installs the responder address as downstream peer state information after verifying the Query Cookie in the GIMPS-response. The responding node can install the querying address as upstream peer state information at two points in time:

1. after the receipt of the initial GIMPS-query, or

2. after a third message in the downstream direction containing the Responder Cookie.

The detailed constraints on precisely when state information is installed are driven by local policy driven by security considerations on prevention of denial-of-service attacks and state poisoning attacks, which are discussed further in Section 7.

Setup of messaging associations begins when both downstream peer addressing information is available and a new messaging association is actually needed. (In many cases, the GIMPS-response message above will identify a downstream peer for whom an appropriate messaging association already exists, in which case no further action is needed.) Setup of the messaging association always starts from the

upstream node, but it can be used equally in both directions. The negotiation of what protocols to use for the messaging association is controlled by the Stack Proposal information exchanged, and the processing is outline in Section 6.6.

Refresh and expiration of all types of state is controlled by timers. State in the routing table has a per-flow, per-direction timer, which expires after a routing state lifetime. It is the responsibility of the querying node to generate a GIMPS-query message, optionally with a Discover-Query payload, before this timer expires, if it believes that the flow is still active. Receipt of the message at the responding node will refresh upstream peer addressing state, and receipt of a GIMPS-response at the querying node will refresh any downstream peer addressing state if it exists. Note that nodes do not control the refresh of upstream peer state themselves, they are dependent on the upstream peer for this.

Messaging associations can be managed by either end. Management consists of tearing down unneeded associations. Whether an association is needed is a local policy decision, which could take into account the cost of keeping the messaging association open, the level of past activity on the association, and the likelihood of future activity (e.g. if there are flows still in place which might generate messages that would use it). Messaging associations can

always be set up on demand, and messaging association status is not made directly visible outside the GIMPS layer. Therefore, even if GIMPS tears down and later re-establishes a messaging association, signaling applications cannot distinguish this from the case where the association is kept permanently open.

5. Message Formats and Encapsulations

5.1 GIMPS Messages

All GIMPS messages begin with a common header, which includes a version number, information about message type, signaling application, and additional control information. The remainder of the message is encoded in an RSVP-style format, i.e., as a sequence of type-length-value (TLV) objects. This subsection describes the possible GIMPS messages and their contents at a high level; a more detailed description of each information element is given in Section 5.2.

The following gives the syntax of GIMPS messages in ABNF [3].

GIMPS-message: A message is either a datagram mode message or a connection mode message. GIMPS can detect which by the encapsulation the message arrives over.

```
GIMPS-message = D-message / C-message
```

D-message: A datagram mode message is either upstream or downstream (slightly different contents are allowed); the common header contains a flag to say which.

```
D-message = D-upstream-message / D-downstream-message
```

C-message: A connection mode message is either upstream or downstream (again, slightly different contents are allowed); the common header contains a flag to say which. Note that upstream and downstream messages can be mixed on a single messaging association.

```
C-message = C-upstream-message / C-downstream-message
```

D-downstream-message: A downstream datagram mode message is used for the GIMPS-query and final handshake in the discovery procedure, and can also be used simply for carrying NSLP data. Note that the Common-Header includes a flag to indicate whether an explicit response is required.

```
D-downstream-message = Common-Header
                       Message-Routing-Information
                       Node-Addressing
                       Session-Identification
```

```
                              [ Stack-Proposal ]
                              [ Query-Cookie / Responder-Cookie ]
                              [ Routing-State-Lifetime ]
                              [ NSLP-Data ]
```


D-upstream-message: An upstream datagram mode message is used for the GIMPS-response in the discovery procedure, and can also be used simply for carrying NSLP data.

```
    D-upstream-message   = Common-Header
                           Message-Routing-Information
                           Node-Addressing
                           Session-Identification
                           [ Stack-Proposal ]
                           [ Query-Cookie [ Responder-Cookie ] ]
                           [ Routing-State-Lifetime ]
                           [ NSLP-Data ]
```

C-downstream-message: A downstream connection mode message is used primarily for carrying NSLP data, but can also be used for the final handshake during discovery. Connection mode messages do not carry node addressing, since this can be inferred from the messaging association.

```
    C-downstream-message = Common-Header
                           Message-Routing-Information
                           Session-Identification
                           [ Responder-Cookie ]
                           [ Routing-State-Lifetime ]
                           [ NSLP-Data ]
```

C-upstream-message: An upstream connection mode message is used primarily for carrying NSLP data, but can also be used for the GIMPS-response during discovery (which is the only case where a Stack-Proposal TLC can be included).

```
C-upstream-message    = Common-Header
                        Message-Routing-Information
                        Session-Identification
                        [ Stack-Proposal ]
                        [ Query-Cookie [ Responder-Cookie ] ]
                        [ Routing-State-Lifetime ]
                        [ NSLP-Data ]
```

5.2 Information Elements

This section describes the content of the various information elements that can be present in each GIMPS message, both the common header, and the individual TLVs. The format description in terms of bit patterns is provided (in an extremely preliminary form) in Appendix C.

5.2.1 The Common Header

Each message begins with a fixed format common header, which contains the following information:

Version: The version number of the GIMPS protocol.

Length: The number of TLVs following in this message.

Signaling application identifier (NSLPID): This describes the specific signaling application, such as resource reservation or firewall control.

GIMPS hop counter: A hop counter to prevent a message from looping

indefinitely.

U/D flag: A bit to indicate if this message is to propagate upstream or downstream relative to the flow.

Response requested flag: A bit to indicate that this message contains a cookie which must be echoed in the response.

5.2.2 TLV Objects

All data following the common header is encoded as a sequence of type-length-value objects. Currently, each object can occur at most once; the set of required and permitted objects is determined by the message type and further information in the common header.

These items are contained in each GIMPS message:

Message-Routing-Information (MRI): Information sufficient to define the route that the flow will take through the network.

```
Message-Routing-Information = message-routing-method
                              method-specific-information
```

The format of the method-specific-information depends on the message-routing-method requested by the signaling application. In the basic path-coupled case, it is just the Flow Identifier as in [22]. Minimally, this could just be the flow destination address; however, to account for policy based forwarding and other issues a more complete set of header fields should be used (see Section 6.2 and Section 6.3 for further discussion).

```
Flow-Identifier = network-layer-version
                  source-address prefix-length
```

```
                        destination-address prefix-length
                        IP-protocol
                        traffic-class
                        [ flow-label ]
                        [ ipsec-SPI / L4-ports]
```

Additional control information defines whether the flow-label, SPI and port information are present, and whether the IP-protocol and traffic-class fields should be interpreted as significant.

Session-Identification (SID): The GIMPS session identifier is a long, cryptographically random identifier chosen by the node which begins the signaling exchange (the signaling application at the node may specify it explicitly, or leave it up to GIMPS to select a value). The length is open, but 128 bits should be more than sufficient to make the probability of collisions orders of magnitude lower than other failure reasons. The session identifier should be considered immutable end-to-end along the flow path (GIMPS never changes it, and signaling applications should propagate it unchanged on messages for the same session).

The following items are optional:

Node addressing: Minimally, this is the IP address at which the GIMPS node originating the message can be reached; this will be used to fill in peer routing state. It may also include a logical interface identifier to assist in route change handling, see Section 6.1, and port and other information relevant to the messaging association protocols. This field must be considered mutable to allow for NAT traversal. The level of flexibility required in this field is discussed in Section 8.5.

Stack Proposal: This field contains information about which combinations of transport and security protocols are proposed for use in messaging associations. This field must be considered immutable between GIMPS peers; see Section 6.6 for further details.

Query-Cookie/Responder-Cookie: A query-cookie is optional in a GIMPS-query message and if present must be echoed in a

GIMPS-response; a response-cookie is optional in a GIMPS-response message, and if present must be echoed in the following downstream message. Cookies are variable length (chosen by the cookie generator) and need to be designed so that a node can determine the validity of a cookie without keeping state. A future version of this specification will include references to techniques for generating such cookies.

Routing-State-Lifetime: The lifetime of GIMPS routing state in the absence of refreshes, measured in seconds. Defaults to 30 seconds.

NSLP-Data: The NSLP payload to be delivered to the signaling application. GIMPS does not interpret the payload content.

5.3 Encapsulation in Datagram Mode

Encapsulation in datagram mode is simple. The complete set of GIMPS payloads for a single message is concatenated together with the common header, and placed in the data field of a UDP datagram. UDP checksums should be enabled. Upstream messages are directly addressed to the adjacent peer. Downstream messages are addressed using information from the Message-Routing-Information and encapsulated with a Router Alert Option. Open issues about alternative encapsulations, addressing possibilities, and router alert option value-field setting are discussed in Section 8.2, Section 8.3 and Section 8.4 respectively.

The source UDP port is selected by the message sender. A destination UDP port should be allocated by IANA. Note that GIMPS may send messages addressed as {flow sender, flow receiver} which could make their way to the flow receiver even if that receiver were GIMPS-unaware. This should be rejected (with an ICMP message) rather than delivered to the user application (which would be unable to use

the source address to identify it as not being part of the normal data flow). Therefore, a "well-known" port would seem to be required.

For the case of basic path-coupled signaling where the MRI information is the Flow Identifier, it is vital that the D-mode message truly mimics the actual data flow, since this is the basis of how the signaling message is attached to the data path. To this end, GIMPS may set the traffic class and (for IPv6) flow label to match the values in the Flow-Identifier if this would be needed to ensure correct routing. Similar considerations may apply to other message routing methods if defined.

5.4 Encapsulation in Connection Mode

Encapsulation in connection mode is more complex, because of the variation in available transport functionality. This issue is treated in Section 5.4.1. The actual encapsulation is given in Section 5.4.2.

5.4.1 Choice of Transport Protocol

It is a general requirement of the NTLP defined in [22] that it should be able to support bundling (of small messages), fragmentation (of large messages), and message boundary delineation. Not all transport protocols natively support all these features.

SCTP [6] satisfies all requirements. (The bundling requirement is met implicitly by the use of Nagle-like algorithms inside the SCTP stack.)

DCCP [7] is message based but does not provide bundling or

fragmentation. Bundling can be carried out by the GIMPS layer sending multiple messages in a single datagram; because the common header includes length information (number of TLVs), the message boundaries within the datagram can be discovered during parsing. Fragmentation of GIMPS messages over multiple datagrams should be avoided, because of amplification of message loss rates that this would cause.

TCP provides both bundling and fragmentation, but not message boundaries. However, the length information in the common header allows the message boundary to be discovered during parsing.

UDP can be augmented as in the DCCP case. (An additional reason for avoiding fragmentation is the lack of congestion control functionality in UDP.)

It can be seen that all of these protocol options can be supported by the basic GIMPS message format already presented. GIMPS messages requiring fragmentation must be carried using a reliable transport protocol, TCP or SCTP.

5.4.2 Encapsulation Format

The GIMPS message, consisting of common header and TLVs, is carried directly in the transport protocol (possibly incorporating transport layer security protection). Further GIMPS messages can be carried in a continuous stream (for TCP), or up to the next transport layer message boundary (for SCTP/DCCP/UDP). This situation is shown in Figure 5; it applies to both upstream and downstream messages.

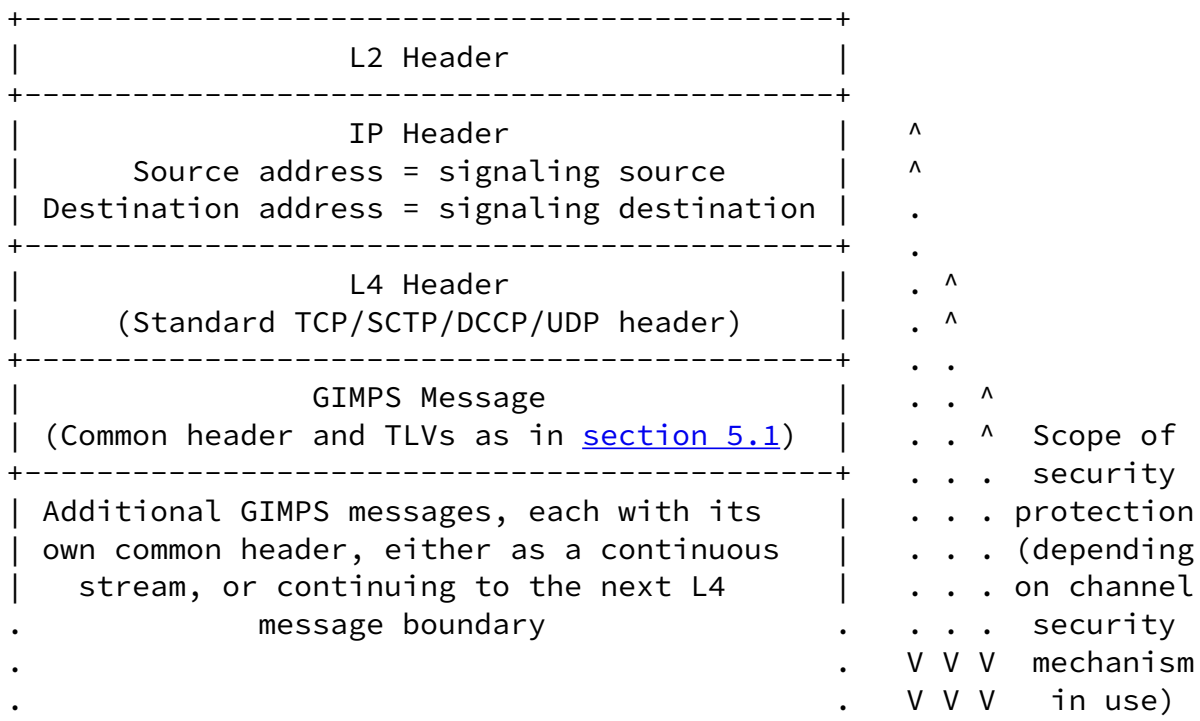


Figure 5: Connection Mode Encapsulation

Note that when GIMPS messages are carried in connection mode in this way, between the GIMPS peers they are treated just like any other traffic by intermediate routers. Indeed, it would be impossible for intermediate routers to carry out any processing on the messages without terminating the transport and security protocols used.

Signaling messages are only ever delivered between peers established in GIMPS-query/response exchanges. Any route change is not detected until another GIMPS-query/response procedure takes place; in the meantime, signaling messages are misdelivered. GIMPS is responsible for prompt detection of route changes to minimise the period during which this can take place.

6. Advanced Protocol Features

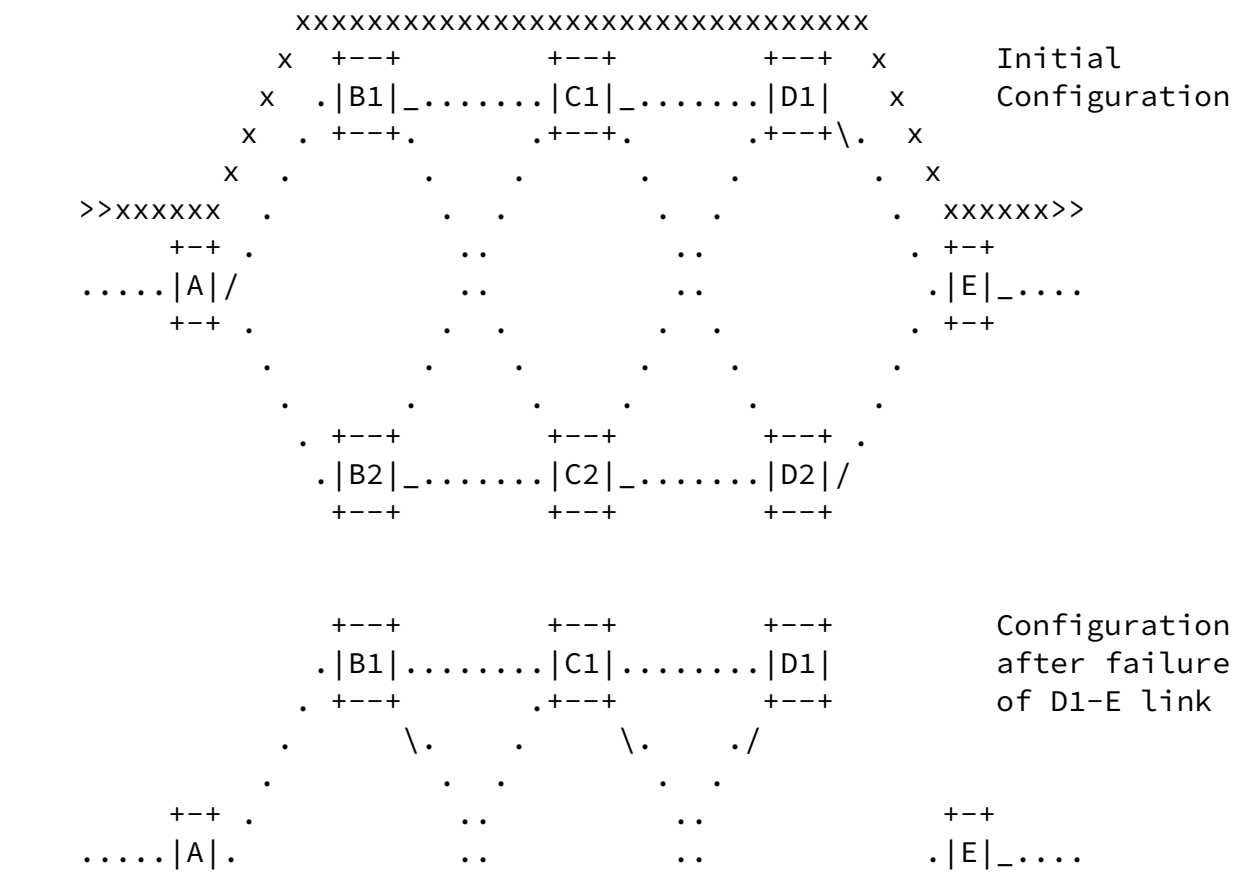
6.1 Route Changes and Local Repair

6.1.1 Introduction

When re-routing takes place in the network, GIMPS and signaling application state needs to be updated for all flows whose paths have changed. The updates to signaling application state are usually signaling application dependent: for example, if the path characteristics have actually changed, simply moving state from the old to the new path is not sufficient. Therefore, GIMPS cannot carry out the complete path update processing. Its responsibilities are to detect the route change, update its own routing state consistently, and inform interested signaling applications at affected nodes.

Route change management is complicated by the distributed nature of the problem. Consider the re-routing event shown in Figure 6. An external observer can tell that the main responsibility for controlling the updates will probably lie with nodes A and E; however, D1 is best placed to detect the event quickly at the GIMPS level, and B1 and C1 could also attempt to initiate the repair.

On the assumption that NSLPs are soft-state based and operate end to end, and because GIMPS also periodically updates its picture of routing state, route changes will eventually be repaired automatically. However, especially if NSLP refresh times are extended to reduce signaling load, the duration of inconsistent state may be very long indeed. Therefore, GIMPS includes logic to deliver prompt notifications to NSLPs, to allow NSLPs to carry out local repair if possible.


Initial
Configuration
Configuration
after failure
of D1-E link

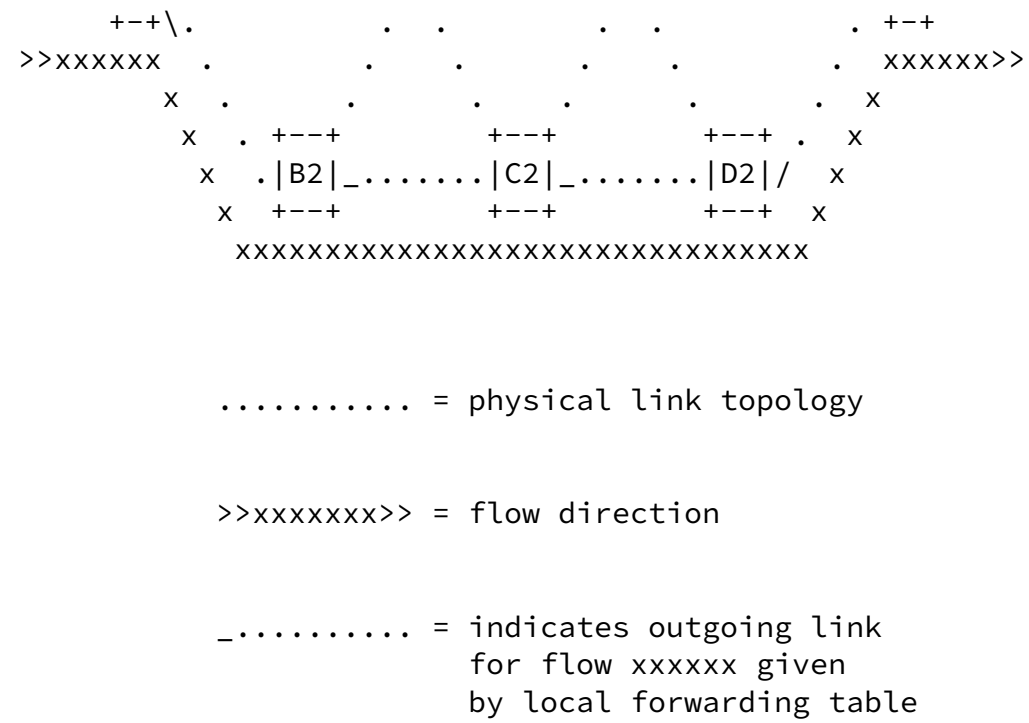


Figure 6: A Re-Routing Event

6.1.2 Route Change Detection

There are two aspects to detecting a route change at a single node:

- Detecting that the downstream path has (or may have) changed.

- Detecting that the upstream path has (or may have) changed (in which case the node may no longer be on the path at all).

At a single node, these processes are largely independent, although clearly a change in downstream path at a node corresponds to a change

in upstream path at the downstream peer. Note that there are two possible aspects of route change:

Interface: The interface through which a flow leaves or enters a node may change.

Peer: The adjacent upstream peer or downstream peer may change.

In general, a route change could include one or the other or both. (In theory it could include neither, although such changes are hard to detect and even harder to do anything useful about.)

There are five mechanisms for a GIMPS node to detect that a route change has occurred, which are listed below. They apply differently depending on whether the change is in the upstream or downstream path, and these differences are summarised in the following table.

Local Trigger: In trigger mode, a node finds out that the next hop has changed. This is the RSVP trigger mechanism where some form of notification mechanism from the routing table to the protocol handler is assumed. Clearly this only works if the routing change is local, not if the routing change happens somewhere a few routing hops away (including the case that the change happens at a GIMPS-unaware node).

Extended Trigger: An extended trigger, where the node checks a link-state routing table to discover that the path has changed. This makes certain assumptions on consistency of route computation (but you probably need to make those to avoid routing loops) and only works within a single area for OSPF and similar link-state protocols. Where available, this offers the most accurate and expeditious indication of route changes, but requires more access to the routing internals than a typical OS may provide.

GIMPS C-mode Monitoring: A node may find that C-mode packets are arriving (from upstream or downstream peer) with a different TTL or on a different interface. This provides no direct information about the new flow path, but indicates that routing has changed and that rediscovery may be required.

Data Plane Monitoring: The signaling application on a node may detect a change in behaviour of the flow, such as TTL change, arrival on a different interface, or loss of the flow altogether. The

signaling application on the node is allowed to notify this information locally to GIMPS.

GIMPS D-mode Probing: In probing mode, each GIMPS node periodically repeats the discovery (GIMPS-query/GIMPS-response) operation. The querying node will discover the route change by a modification in the Node-Addressing information in the GIMPS-response. This is similar to RSVP behavior, except that there is an extra degree of freedom since not every message needs to repeat the discovery, depending on the likely stability of routes. All indications are that, leaving mobility aside, routes are stable for hours and days, so this may not be necessary on a 30-second interval, especially if the other techniques listed above are available.

When these methods discover a route change in the upstream direction, this cannot be handled directly by GIMPS at the detecting node, since route discovery proceeds only in the downstream direction. Therefore, to exploit these mechanisms, it must be possible for GIMPS to send a notification message in the upstream direction to initiate this. (This would be possible for example by setting an additional flag in the Common-Header of an upstream message.)

Method	Downstream	Upstream
Local Trigger	Discovers new downstream interface (and peer if local)	Not applicable
Extended Trigger	Discovers new downstream interface and may determine new downstream peer	May determine that route from upstream peer will have changed
C-Mode Monitoring	Provides hint that change has occurred	Provides hint that change has occurred

Data Plane Monitoring	Not applicable	NSLP informs GIMPS that a change may have occurred
D-Mode Probing	Discovers changed Node-Addressing in GIMPS-response	Discovers changed Node-Addressing in GIMPS-query

6.1.3 Local Repair

Once a node has detected that a change may have occurred, there are three possible cases:

1. Only an upstream change is indicated. There is nothing that can be done locally; GIMPS must propagate a notification to its upstream peer.

2. A downstream change has been detected and an upstream change cannot be ruled out. Although some local repair may be appropriate, it is difficult to decide what, since the path change may actually have taken place upstream of the detecting node (so that this node is no longer on the path at all).

3. A downstream change has been detected, but there is no upstream change. In this case, the detecting node is the true crossover router, i.e. the point in the network where old and new paths diverge. It is the correct node to initiate the local repair process.

In case (3), i.e. at the upstream crossover node, the local repair

process is initiated by the GIMPS level as follows:

- GIMPS marks its downstream routing state information for this flow as 'invalid', unless the route change was actually detected by D-mode probing (in which case the new state has already been installed).

- GIMPS notifies the local NSLP that local repair is necessary.

It is assumed that the second step will typically trigger the NSLP to generate a downstream message, and the attempt to send it will stimulate a GIMPS-query/response. This signaling application message will propagate downstream, also discovering the new route, until it rejoins the old path; the node where this happens may also have to carry out local repair actions.

A problem is that there is usually no robust technique to distinguish case (2) from case (3), because of the relative weakness of the techniques in determining that upstream change has not occurred. (They can be effective in determining that a change has occurred; however, even where they can tell that the route from the upstream peer has not changed, they cannot rule out a change beyond that peer.) There is therefore a danger that multiple nodes within the network would attempt to carry out local repair in parallel.

One possible technique to address this problem is that a GIMPS node

that detects case (3) locally, rather than initiating local repair immediately, still sends a route change notification upstream, just in case (2) actually applies. If the upstream peer locally detects no downstream route change, it can signal this to the downstream node (e.g. by setting another flag in the Common-Header of a GIMPS message). This acts to damp the possibility of a 'local repair storm', at the cost of an additional peer-peer round trip time.

6.1.4 Local Signaling Application State Removal

After a route change, a signaling application may wish to remove state at another node which is no longer on the path. However, since it is no longer on the path, in principle GIMPS can no longer send messages to it. (In general, provided this state is soft, it will time out anyway; however, the timeouts involved may have been set to be very long to reduce signaling load.) The requirement to remove state in a specific peer node is identified in [25].

This requirement can be met provided that GIMPS is able to 'remember' the old path to the signaling application peer for the period while the NSLP wishes to be able to use it. Since NSLP peers are a single GIMPS hop apart, the necessary information is just the old entry in the node's routing state table for that flow. Rather than requiring the GIMPS level to maintain multiple generations of this information, it can just be provided to the signaling application in the same node (in an opaque form), which can store it if necessary and provide it back to the GIMPS layer in case it needs to be used. This information is denoted as 'SII-Handle' in the abstract API of Appendix D; however, the details are an implementation issue which do not affect the rest of the protocol.

6.1.5 Operation with Heterogeneous NSLPs

A potential problem with route change detection is that the detecting GIMPS node may not implement all the signaling applications that need to be informed. Therefore, it would need to be able to send a notification back along the unchanged path to trigger the nearest signaling application aware node to take action. If multiple signaling applications are in use, it would be hard to define when to stop propagating this notification. However, given the rules on message interception and routing state maintenance in Section 4.2, Section 4.3 and Section 8.4, this situation cannot arise: all NSLP peers are exactly one GIMPS hop apart.

The converse problem is that the ability of GIMPS to detect route changes by purely local monitoring of forwarding tables is more limited. (This is probably an appropriate limitation of GIMPS functionality. If we need a protocol for distributing notifications

about local changes in forwarding table state, a flow signaling protocol is probably not the right starting point.)

6.2 Policy-Based Forwarding and Flow Wildcarding

Signaling messages almost by definition need to contain address and port information to identify the flow they are signaling for. We can divide this information into two categories:

Message-Routing-Information: This is the information needed to determine how a message is routed within the network. It may include a number of flow N-tuple parameters, and is carried as an object in each GIMPS message (see Section 5.1).

Additional Packet Classification Information: This is any further higher layer information needed to select a subset of packets for special treatment by the signaling application. The need for this is highly signaling application specific, and so this information is invisible to GIMPS (if indeed it exists); it will be carried only in the corresponding NSLP.

The correct pinning of signaling messages to the data path depends on how well the downstream messages in datagram mode can be made to be routed correctly. Two strategies are used:

The messages themselves match the flow in destination address and possibly other fields (see Section 5.3 and Section 8.3 for further discussion). In many cases, this will cause the messages to be routed correctly even by GIMPS-unaware nodes.

A GIMPS-aware node carrying out policy based forwarding on higher layer identifiers (in particular, the protocol and port numbers for IPv4) should take into account the entire Message-Routing-Information object in selecting the outgoing interface rather than relying on the IP layer.

The current Message-Routing-Information format allows a limited

degree of 'wildcarding', for example by applying a prefix length to the source or destination address, or by leaving certain fields unspecified. A GIMPS-aware node must verify that all flows matching the Message-Routing-Information would be routed identically in the downstream direction, or else reject the message with an error.

6.3 NAT Traversal

As already noted, GIMPS messages must carry packet addressing and higher layer information as payload data in order to define the flow signalled for. (This applies to all GIMPS messages, regardless of

how they are encapsulated or which direction they are travelling in.) At an addressing boundary the data flow packets will have their headers translated; if the signaling payloads are not likewise translated, the signaling messages will refer to incorrect (and probably meaningless) flows after passing through the boundary.

The simplest solution to this problem is to require that a NAT is GIMPS-aware, and to allow it to modify datagram mode messages based on the contents of the Message-Routing-Information payload. (This is making the implicit assumption that NATs only rewrite the header fields included in this payload, and not higher layer identifiers.) Provided this is done consistently with the data flow header translation, signaling messages will be valid each side of the boundary, without requiring the NAT to be signaling application aware. An outline of the set of operations necessary on a downstream datagram mode message is as follows:

1. Verify that bindings for the data flow are actually in place.

2. Create bindings for subsequent C-mode signaling (based on the information in the Node-Addressing field).

3. Create a new Message-Routing-Information payload with fields modified according to the data flow bindings.

4. Create a new Node-Addressing payload with fields to force upstream D-mode messages through the NAT, and to allow C-mode exchanges using the C-mode signaling bindings.

5. Forward the message with these new payloads.

The original Message-Routing-Information and Node-Addressing payloads should be retained in the message, but encapsulated in a new TLV type. (In the case of a sequence of NATs, this TLV would become a list.) This TLV essentially becomes a recorded route for the D-mode message; a GIMPS node that wished to do topology hiding could replace these original payloads with opaque tokens, or omit them altogether. Note that a consequence of this approach is that the routing state tables at the actual signaling application peers (either side of the NAT) are no longer directly compatible (in particular, the values of Message-Routing-Information are different.

The case of traversing a GIMPS unaware NAT is for further study. There is a dual problem of whether the GIMPS peers either side of the boundary can work out how to address each other, and whether they can work out what translation to apply to the Message-Routing-Information from what is done to the signaling packet headers. The fundamental problem is that GIMPS messages contain 3 or 4 interdependent

addresses which all have to be consistently translated, and existing generic NAT traversal techniques such as STUN [21] can process only two.

6.4 Interaction with IP Tunnelling

The interaction between GIMPS and IP tunnelling is very simple. An

IP packet carrying a GIMPS message is treated exactly the same as any other packet with the same source and destination addresses: in other words, it is given the tunnel encapsulation and forwarded with the other data packets.

Tunnelled packets will not be identifiable as GIMPS messages until they leave the tunnel, since any router alert option and the standard GIMPS protocol encapsulation (e.g. port numbers) will be hidden behind the standard tunnel header. If signaling is needed for the tunnel itself, this has to be initiated as a separate signaling session by one of the tunnel endpoints - that is, the tunnel counts as a new flow. Because the relationship between signaling for the 'microflow' and signaling for the tunnel as a whole will depend on the signaling application in question, we are assuming that it is a signaling application responsibility to be aware of the fact that tunnelling is taking place and to carry out additional signaling if necessary; in other words, one tunnel endpoint must be signaling application aware.

In some cases, it is the tunnel exit point (i.e. the node where tunnelled data and downstream signaling packets leave the tunnel) that will wish to carry out the tunnel signaling, but this node will not have knowledge or control of how the tunnel entry point is carrying out the data flow encapsulation. This information could be carried as additional data (an additional GIMPS payload) in the tunnelled signaling packets if the tunnel entry point was at least GIMPS aware. This payload would be the GIMPS equivalent of the RSVP SESSION_ASSOC object of [12]. Whether this functionality should really be part of GIMPS and if so how the payload should be handled will be considered in a later version.

6.5 IPv4-IPv6 Transition and Interworking

GIMPS itself is essentially IP version neutral (version dependencies are isolated in the formats of the Message-Routing-Information and Node-Addressing TLVs, and GIMPS also depends on the version independence of the protocols that support messaging associations). In mixed environments, GIMPS operation will be influenced by the IP transition mechanisms in use. This section provides a high level overview of how GIMPS is affected, considering only the currently predominant mechanisms.

Dual Stack: (This applies both to the basic approach described in [26] as well as the dual-stack aspects of more complete architectures such as [28].) In mixed environments, GIMPS should use the same IP version as the flow it is signaling for; hosts which are dual stack for applications and routers which are dual stack for forwarding should have GIMPS implementations which can support both IP versions.

In theory, for some connection mode encapsulation options, a single messaging association could carry signaling messages for flows of both IP versions, but the saving seems of limited value. The IP version used in datagram mode is closely tied to the IP version used by the data flow, so it is intrinsically impossible for a IPv4-only or IPv6-only GIMPS node to support signaling for flows using the other IP version.

Applications with a choice of IP versions might select a version for which GIMPS support was available in the network, which could be established by running parallel discovery procedures. In theory, a GIMPS message related to a flow of one IP version could flag support for the other; however, given that IPv4 and IPv6 could easily be separately routed, the correct GIMPS peer for a given flow might well depend on IP version anyway, making this flagged information irrelevant.

Packet Translation: (Applicable to SIIT [5] and NAT-PT [13].) Some transition mechanisms allow IPv4 and IPv6 nodes to communicate by placing packet translators between them. From the GIMPS perspective, this should be treated essentially the same way as any other NAT operation (e.g. between 'public' and 'private' addresses) as described in Section 6.3. In other words, the translating node needs to be GIMPS aware; it will run GIMPS with IPv4 on some interfaces and with IPv6 on others, and will have to translate the Message-Routing-Information payload between IPv4 and IPv6 formats for flows which cross between the two. The translation rules for the fields in the payload (including e.g. traffic class and flow label) are as defined in [5].

Tunnelling: (Applicable to 6to4 [15] and a whole host of other tunnelling schemes.) Many transition mechanisms handle the problem of how an end to end IPv6 (or IPv4) flow can be carried over intermediate IPv4 (or IPv6) regions by tunnelling; the methods

tend to focus on minimising the tunnel administration overhead.

From the GIMPS perspective, the treatment should be as similar as possible to any other IP tunnelling mechanism, as described in Section 6.4. In particular, the end to end flow signaling will pass transparently through the tunnel, and signaling for the

tunnel itself will have to be managed by the tunnel endpoints. However, additional considerations may arise because of special features of the tunnel management procedures. For example, [16] is based on using an anycast address as the destination tunnel endpoint. It might be unwise to carry out signaling for the tunnel to such an address, and the GIMPS implementation there would not be able to use it as a source address for its own signaling messages (e.g. GIMPS-responses). Further analysis will be contained in a future version of this specification.

6.6 Messaging Association Protocol Negotiation

A key attribute of GIMPS is that it is flexible in its ability to use existing transport and security protocols. Different transport protocols may have performance attributes appropriate to different environments; different security protocols may fit appropriately with different authentication infrastructures. Even if a single choice for GIMPS could be agreed today, the need to support new protocols in the future cannot be ruled out. Therefore, some sort of protocol negotiation capability is required.

The implicit requirements for protocol negotiation are as follows:

- It should be possible to request a set of protocols (e.g. TLS/TCP or SCTP/IPsec), not just a single protocol.

- The negotiation should complete in 1 RTT.

- The negotiation should be resistant to bidding-down ("man in the middle") attacks.

- At the same time, the message elements involved should allow NAT traversal.

- The set of possible protocols should be extensible.

The stacking requirements are reminiscent of [10], and the negotiation requirements of [20], and the following outline is based on the same principles. In particular, the latter should be read for a more detailed security discussion.

- Each possible "protocol layer" is represented by an IANA-assigned tag. A protocol layer defines a well-known protocol (such as "TCP") and a set of rules for its use (such as "Connect from Querying Node").

- A protocol layer may define some security related parameters, and

will probably also define some addressing options; the latter would be carried in the Node-Addressing TLV and are not considered further here.

- A "profile" is a sequence of protocol layers.

- A Stack-Proposal TLV consists of a sequence of profiles, and any associated security parameters.

- When attempting to set up a messaging association, a node includes a Stack-Proposal TLV in the GIMPS-query. The contents of the TLV must be fixed for a given outbound interface and NSLPID.

- The responding node includes another Stack-Proposal in the GIMPS-Response. The contents of this TLV must also be fixed for a given outbound interface and NSLPID.

- The querying node selects a common profile from the proposals and sets up the protocol layers accordingly. Once the messaging association is open, it repeats the Stack-Proposal from the GIMPS-Response. The responding node can verify this to ensure that no bidding down attack has occurred.

The exchanges parallel the cookie exchanges which protect routing state setup, but they are largely independent. (The cookie exchanges can be used to protect nodes from denial of service attacks masquerading as a messaging association protocol setup, in case the connection setup procedure for one of the protocols is DoS-vulnerable, as is the case with TCP.)

It is expected that the initial set of protocol layers will be very small; however, it could be extended, e.g. to allow for different configurations (e.g. "Connect from Responding Node" or requiring the use of particular protocol options) or entirely new protocols.

7. Security Considerations

The security requirement for the GIMPS layer is to protect the signaling plane against identified security threats. For the signaling problem as a whole, these threats have been outlined in [23]; the NSIS framework [22] assigns a subset of the responsibility to the NTLP. The main issues to be handled can be summarised as:

Message Protection: Signaling message content should be protected against eavesdropping, modification, injection and replay while in transit. This applies both to GIMPS payloads, and GIMPS should also provide such protection as a service to signaling applications between adjacent peers.

State Integrity Protection: It is important that signaling messages are delivered to the correct nodes, and nowhere else. Here, 'correct' is defined as 'the appropriate nodes for the signaling given the Message-Routing-Information'. In the case where the MRI is the Flow Identification for path-coupled signalling, 'appropriate' means 'the same nodes that the infrastructure will route data flow packets through'. (GIMPS has no role in deciding whether the data flow itself is being routed correctly; all it can do is ensure the signaling is routed consistently with it.) GIMPS uses internal state to decide how to route signaling messages, and this state needs to be protected against corruption.

Prevention of Denial of Service Attacks: GIMPS nodes and the network have finite resources (state storage, processing power, bandwidth). The protocol should try to minimise exhaustion attacks against these resources and not allow GIMPS nodes to be used to launch attacks on other network elements.

The main missing issue is handling authorisation for executing signaling operations (e.g. allocating resources). This is assumed to be done in each signaling application.

In many cases, GIMPS relies on the security mechanisms available in messaging associations to handle these issues, rather than introducing new security measures. Obviously, this requires the interaction of these mechanisms with the rest of the GIMPS protocol

to be understood and verified, and some aspects of this are discussed in Section 6.6.

7.1 Message Confidentiality and Integrity

GIMPS can use messaging association functionality, such as TLS or IPsec, to ensure message confidentiality and integrity. In many cases, confidentiality of GIMPS information itself is not likely to

be a prime concern, in particular since messages are often sent to parties which are unknown ahead of time, although the content visible even at the GIMPS level gives significant opportunities for traffic analysis. Signaling applications may have their own mechanism for securing content as necessary; however, they may find it convenient to rely on protection provided by messaging associations, particularly if this is provided efficiently and if it runs unbroken between signaling application peers.

7.2 Peer Node Authentication

Cryptographic protection (of confidentiality or integrity) requires a security association with session keys, which can be established during an authentication and key exchange protocol run based on shared secrets, public key techniques or a combination of both. Authentication and key agreement is possible using the protocols associated with the messaging association being secured (TLS incorporates this functionality directly; IKE, IKEv2 or KINK can provide it for IPsec). GIMPS nodes rely on these protocols to authenticate the identity of the next hop, and GIMPS has no authentication capability of its own.

However, with discovery, there are few effective ways to know what is the legitimate next or previous hop as opposed to an impostor. In other words, cryptographic authentication here only provides assurance that a node is 'who' it is (i.e. the legitimate owner of

identity in some namespace), not 'what' it is (i.e. a node which is genuinely on the flow path and therefore can carry out signaling for a particular flow). Authentication provides only limited protection, in that a known peer is unlikely to lie about its role. Additional methods of protection against this type of attack are considered in Section 7.3 below.

It is open whether peer node authentication should be made signaling application dependent; for example, whether successful authentication could be made dependent on presenting authorisation to act in a particular signaling role (e.g. signaling for QoS). The abstract API of Appendix D allows GIMPS to forward such policy and authentication decisions to the NSLP it is serving.

7.3 Routing State Integrity

The internal state in a node (see Section 4.1), specifically the upstream and downstream peer identification, is used to route messages. If this state is corrupted, signaling messages may be misdirected.

In the case where the message routing method is path-coupled

signaling, the messages need to be routed identically to the data flow described by the Flow Identifier, and the routing state table is the GIMPS view of how these flows are being routed through the network in the immediate neighbourhood of the node. Routes are only weakly secured (e.g. there is usually no cryptographic binding of a flow to a route), and there is no other authoritative information about flow routes than the current state of the network itself. Therefore, consistency between GIMPS and network routing state has to be ensured by directly interacting with the routing mechanisms to ensure that the upstream and downstream signaling peers are the appropriate ones for any given flow. A good overview of security issues and techniques in this sort of context is provided in [27].

Downstream peer identification is installed and refreshed only on receiving a GIMPS-reponse message (compare Figure 4). This must echo the cookie from a previous GIMPS-query message, which will have been sent downstream along the flow path (in datagram mode, i.e. end-to-end addressed). Hence, only the true next peer or an on-path attacker will be able to generate such a message, provided freshness of the cookie can be checked at the querying node.

Upstream peer identification can be installed directly on receiving a GIMPS-query message containing addressing information for the upstream peer. However, any node in the network could generate such a message (indeed, almost any node in the network could be the genuine upstream peer for a given flow). To protect against this, two strategies are possible:

Filtering: the receiving node may be able to reject signaling messages which claim to be for flows with flow source addresses which would be ruled out by ingress filtering. An extension of this technique would be for the receiving node to monitor the data plane and to check explicitly that the flow packets are arriving over the same interface and if possible from the same link layer neighbour as the datagram mode signaling packets. (If they are not, it is likely that at least one of the signaling or flow packets is being spoofed.) Signaling applications should only install state on the route taken by the signaling itself.

Authentication (weak or strong): the receiving node may refuse to install upstream state until it has handshaked by some means with the upstream peer. This handshaking could be as simple as requesting the upstream peer to echo the response cookie in the discover-response payload of a GIMPS-response message (to discourage nodes impersonating upstream peers from using forged source addresses); or, it could be full peer authentication within the messaging association, the reasoning being that an authenticated peer can be trusted not to pretend that it is on

path when it is not.

The second technique also plays a role in denial of service prevention, see below. In practice, a combination of both techniques may be appropriate.

7.4 Denial of Service Prevention

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

However, GIMPS can still be subjected to denial-of-service attacks where an attacker using forged source addresses forces a node to establish state without return routability, causing a problem similar to TCP SYN flood attacks. In addition to vulnerabilities of a next peer discovery an unprotected path discovery procedure might introduce more denial of service attacks since a number of nodes could possibly be forced to allocate state. Furthermore, an adversary might modify or replay unprotected signaling messages. 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 GIMPS-capable 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.) Relying only on upper layer security, for example based on CMS, might open a larger door for denial of service attacks since the messages are often only one-shot-messages without utilizing multiple roundtrips and DoS protection mechanisms.

There are at least three defenses against these attacks:

1. The responding node does not establish a session or discover its next hop on receiving the GIMPS-query message, but can wait for a setup message on a reliable channel. If the reliable channel exists, the additional delay is a one one-way delay and the total 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 [6] and other modern protocols.)

3. If there is a chance that the next-hop node 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.

Once a node has decided to establish routing state, there may still be transport and security state to be established between peers. This state setup is also vulnerable to additional denial of service attacks. GIMPS relies on the lower layer protocols that make up messaging associations to mitigate such attacks. The current description assumes that the upstream node is always the one wishing to establish a messaging association, so it is typically the downstream node that needs to be protected. Extensions are considered in Section 8.6; these would require further analysis.

8. Open Issues

8.1 Protocol Naming

Alternate names:
GIST: General Internet Signaling Transport
GIMPS: General Internet Messaging Protocol for Signaling
LUMPS: Lightweight Universal Messaging for Path associated Signaling

There is a danger of some ambiguity as to whether the protocol name refers to the complete transport stack below the signaling applications, or only to the additional protocol functionality above the standard transport protocols (UDP, TCP etc.) The NSIS framework uses the term NTLP for the first, but this specification uses the GIST/variants names for the second (see Figure 2 in Section 3.1). In other words, this specification proposes to meet the requirements for NTLP functionality by layering GIMPS/... over existing standard transport protocols. It isn't clear if additional terminological surgery is needed to make this clearer.

8.2 General IP Layer Issues

Some NSIS messages have to be addressed end-to-end but intercepted at intermediate routers, and this imposes some special constraints on how they can be encapsulated. RSVPv1 [9] primarily uses raw IP with a specific protocol number (46); a UDP encapsulation is also possible for hosts unable to perform raw network i/o. RSVP aggregation [18] uses an additional protocol number (134) to bypass certain interior nodes.

The critical requirements for the encapsulation at this level are that routers should be able to identify signaling packets for processing, and that they should not mis-identify packets for 'normal' end-to-end user data flows as signaling packets. The current assumption is that UDP encapsulation can be used for such messages, by allocating appropriate (new) value codes for the router alert option (RAO) [1][4] to identify NSIS messages. Specific open issues about how to allocate such values are discussed in Section 8.4.

An alternative approach would be to use raw IP with the RSVP protocol numbers and a new RSVP version number. Although this would provide some more commonality with existing RSVP implementations, the NAT traversal problems for such an encapsulation seem much harder to solve. Specifically, any unmodified NAT (which performed address sharing) would be unable to process any such traffic since they need to understand a higher-layer field (such as TCP or UDP port) to use as a demultiplexer.

8.3 Encapsulation and Addressing for Datagram Mode

The discussion in Section 4 essentially assumes that datagram mode messages are UDP encapsulated. This leaves open the question of whether other encapsulations are possible, and exactly how these messages should be addressed.

As well as UDP/IP (and raw IP as discussed and temporarily ruled out in Section 8.2), DCCP/IP and UDP/IPsec could also be considered as 'datagram' encapsulations. However, they still require explicit addressing between GIMPS peer nodes and some per-peer state to be set up and maintained. Therefore, it seems more appropriate to consider these encapsulation options as possible messaging association types, for use where there is a need for congestion control or security protection but without reliability. This would leave UDP/IP as the single encapsulation allowed for all datagram mode messages.

Addressing for upstream datagram mode messages is simple: the IP source address is the signaling source address, and the IP destination address is the signaling destination address (compare Figure 1). For downstream datagram mode messages, the IP destination address will be the flow destination address, but the IP source address could be either of the flow source address or signaling source address. Some of the relative merits of these options are as follows:

- Using the flow source address makes it more likely that the message will be correctly routed through any intermediate NSIS-unaware region which is doing load sharing or policy routing on the {source, destination} address pair. If the signaling source address is used, the message will be intercepted at some node closer to the flow destination, but it may not be the same as the next node for the data flow packets.

- Conversely, using the signaling source address means that ICMP error messages (specifically, unreachable port or address) will be correctly delivered to the message originator, rather than being sent back to the flow source. Without seeing these messages, it is very difficult for the querying node to recognise that it is the last NSIS node on the path. In addition, using the signaling source address may make it possible to exchange messages through GIMPS unaware NATs (although it isn't clear how useful the resulting messages will be, see Section 6.3).

It is not clear which of these situations it is more important to handle correctly and hence which source addressing option to use. (RSVP uses the flow source address, although this is primarily for multicast routing reasons.) A conservative approach would be to allow

both, possibly even in parallel (although this might open up the protocol to amplification attacks).

8.4 Intermediate Node Bypass and Router Alert Values

We assume that the primary mechanism for intercepting messages is the use of the RAO. The RAO contains a 16 bit value field, within which 35 values have currently been assigned by IANA. It is open how to assign values for use by GIMPS messages to optimise protocol processing, i.e. to minimise the amount of slow-path processing that nodes have to carry out for messages they are not actually interested in the content of.

There are two basic reasons why a GIMPS node might wish to ignore a message:

- because it is for a signaling application that the node does not process;

- because even though the signaling application is present on the node, the interface on which the message arrives is only processing signaling messages at the aggregate level and not for individual flows (compare [18]).

Conversely, note that a node might wish to process a number of different signaling applications, either because it was genuinely multifunctional or because it processed several versions of the same application. (Note from Appendix C.1 that different versions are distinguished by different NSLP identifiers.)

Some or all of this information could be encoded in the RAO value field, which would then allow messages to be filtered on the fast path. There is a tradeoff between two approaches here, whose evaluation depends on whether the processing node is specialised or general purpose:

Fine-Grained: The signaling application (including specific version) and aggregation level are directly identified in the RAO value. A specialised node which handles only a single NSLP can efficiently ignore all other messages; a general purpose node may have to match the RAO value in a message against a long list of possible values.

Coarse-Grained: IANA allocates RAO values for 'popular' applications or groups of applications (such as 'All QoS Signaling Applications'). This speeds up the processing in a general purpose node, but a specialised node may have to carry out further processing on the GIMPS common header to identify the precise

messages it needs to consider.

These considerations imply that the RAO value should not be tied directly to the NSLP id, but should be selected for the application on broader considerations of likely deployment scenarios. Note that the exact NSLP is given in the GIMPS common header, and some implementations may still be able to process it on the fast path. The semantics of the node dropping out of the signaling path are the same however the filtering is done.

There is a special consideration in the case of the aggregation level. In this case, whether a message should be processed depends on the network region it is in (specifically, the link it is on). There are then two basic possibilities:

1. All routers have essentially the same algorithm for which messages they process, i.e. all messages at aggregation level 0. However, messages have their aggregation level incremented on entry to an aggregation region and decremented on exit.

2. Router interfaces are configured to process messages only above a certain aggregation level and ignore all others. The aggregation level of a message is never changed; signaling messages for end to end flows have level 0, but signaling messages for aggregates are generated with a higher level.

The first technique requires aggregating/deaggregating routers to be configured with which of their interfaces lie at which aggregation level, and also requires consistent message rewriting at these boundaries. The second technique eliminates the rewriting, but requires interior routers to be configured also. It is not clear what the right trade-off between these options is.

8.5 Messaging Association Flexibility

Section 4 discusses the use of 0 or 1 messaging associations between any pair of GIMPS nodes. However, logically it would be quite possible to have more than one association, for example:

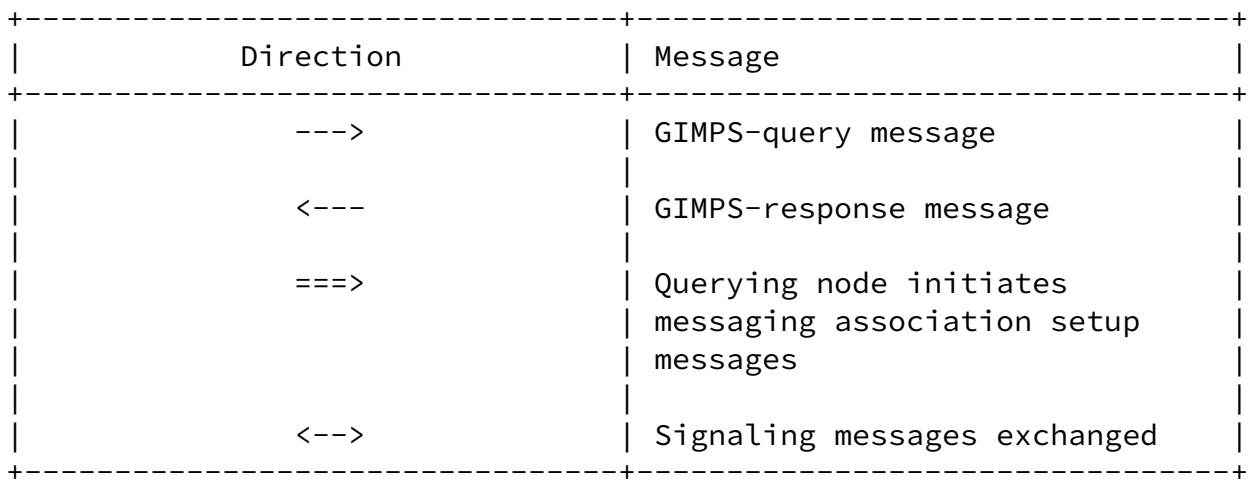
- to allow different reliability characteristics;
- to provide different levels of security protection or to have security separation between different signaling streams;
- even simply to have load split between different connections according to priority (so there could be two associations with identical protocol stacks).

It is possible to imagine essentially infinite flexibility in these options, both in terms of how many possibilities are allowed and how nodes signal their capabilities and preferences, without much changing the overall GIMPS structure. (The GIMPS-query and GIMPS-response messages described in section Section 4.3 can be used

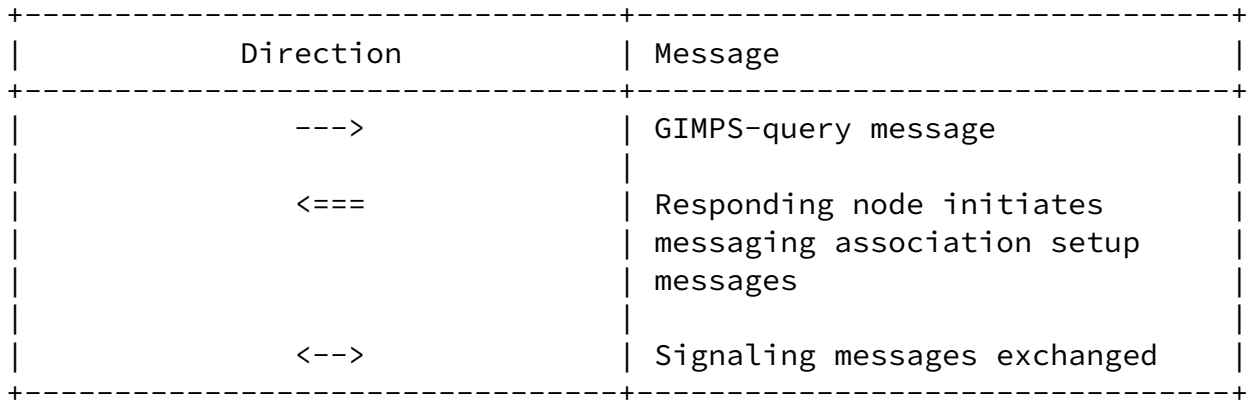
to exchange this information.) What is not clear is how much flexibility is actually needed.

8.6 Messaging Association Setup Message Sequences

The discussion of Section 4.3 assumes a simple fixed message sequence, which we can picture as follows:



There are several variants which could be considered at this level, for example whether the messaging association could be set up by the responding node:



This saves a message but may be vulnerable to additional denial of service attacks.

Another open area is how to manage the protocol exchanges that take place in setting up the messaging association itself. It is probably

an implementation matter to consider whether to carry out, for example, the SCTP 4-way handshake only after IKE exchanges (for IPsec SA initialisation) have completed, or whether these can be done partly in parallel. A more radical step is to carry the initial request and response messages of both exchanges as payloads in the GIMPS-query/response exchange, with the request message initially formatted by the querying node with an unspecified destination IP address. This would require modifications to the protocol implementations (especially at the querying node) similar to what is needed for NAT traversal; it would have to be evaluated whether this was worth the one or two round trip times that are saved. Both this technique and the "reverse connection" approach above can be considered optimisations; given an appropriate negotiation procedure in the base protocol as discussed in Section 6.6 they probably do not need to be considered further for the initial version.

A final area is how the responding node propagates the signaling message downstream. It could initiate the downstream discovery process as soon as it received the initial GIMPS-query message, or it could wait until the first signaling application message has been received (which might not be until a messaging association has been established). A similar timing question applies to when it should initiate its own downstream messaging associations. It is possible that all these options are simply a matter for implementation freedom, although leaving them open will make mobility and re-routing behaviour rather harder to analyse, and again there are denial of service implications for some approaches (see Section 7.4).

8.7 GIMPS Support for Message Scoping

Many signaling applications are interested in sending messages over a specific region of the network. Message scoping of this nature seems to be hard to achieve in a topologically robust way, because such region boundaries are not well defined in the network layer.

It may be that the GIMPS layer can assist such scoping, by detecting

and counting different types of nodes in the signaling plane. The simplest solution would be to count GIMPS nodes supporting the relevant signaling application - this is already effectively done by the GIMPS hop count. A more sophisticated approach would be to track the crossing of aggregation region boundaries, as introduced in Section 8.4. Whether this is plausible depends on the willingness of operators to configure such boundary information in their routers.

8.8 Additional Discovery Mechanisms

The routing state maintenance procedures described in Section 4.3 are strongly focussed on the problem of discovering, implicitly or

explicitly, the neighbouring peers on the flow path - which is the necessary functionality for path-coupled signaling.

As well as the GIMPS-query/response discovery mechanism, other techniques may sometimes also be possible. For example, in many environments, a host has a single access router, i.e. the downstream peer (for outgoing flows) and the upstream peer (for incoming ones) are known a priori. More generally, a link state routing protocol database can be analysed to determine downstream peers in more complex topologies, and maybe upstream ones if strict ingress filtering is in effect. More radically, much of the GIMPS protocol is unchanged if we consider off-path signaling nodes, although there are significant differences in some of the security analysis (Section 7.3). However, none of these possibilities are considered further in this specification.

8.9 Alternative Message Routing Requirements

The initial assumption of GIMPS is that signaling messages are to be routed identically to data flow messages. For this case of path-coupled signaling, the MRI and upstream/downstream flag (in the Common-Header) define the flow and the relationship of the signaling

to it sufficiently for GIMPS to route its messages correctly. However, some additional modes of routing signaling messages have been identified:

Predictive Routing: Here, the intent is to send signaling along a path that the data flow may or will follow in the future. Possible cases are pre-installation of state on the backup path that would be used in the event of a link failure; and predictive installation of state on the path that will be used after a mobile node handover. It is currently unclear whether these cases can be met using the existing GIMPS routing capabilities (and if they cannot, whether they are in the initial scope of the work).

NAT Address Reservations: This applies to the case where a node behind a NAT wishes to use NSIS signaling to reserve an address from which it can be reached by a sender on the other side. This requires a message to be sent outbound from what will be the flow receiver although no reverse routing state exists. One possible solution (assumed in [24]) is to construct a message with the Flow-Routing-Information matching the possible senders and send it as though it was downstream signaling. It is not clear whether signaling for the 'wrong direction' in this way will always be treated consistently by GIMPS, especially if routing policies and encapsulations for inbound and outbound traffic are treated very differently within the rest of the network.

In the current structure of the protocol definition, the way to handle these requirements (if they are needed) is to define a new message routing method which replaces the basic path-coupled version. The requirements for defining a new routing method include the following:

- Defining the format of the MRI for the new message routing method type.

- Defining how D-mode messages should be encapsulated and routed corresponding to this MRI.

- Defining any filtering or other security mechanisms that should be used to validate the MRI in a D-mode message.

- Defining how the MRI format is processed on passing through a NAT.

8.10 Congestion Control in Datagram Mode

The GIMPS-query and GIMPS-response messages may suffer from message loss (e.g. due to congestion or corruption). Because a successful handshake is necessary before a messaging association can even be initiated, GIMPS must provide its own recovery method for these cases. A working assumption is that the querying node can repeat the GIMPS-query with an exponential backoff until a response is received or some retry threshold is reached.

More subtle is the case where there is a stream of D-mode messages with no immediate feedback from the neighbour node. This could be the case where a signaling application was generating messages for stateless processing in the interior of the network. Here, the appropriate approach may be to use rate-limiting algorithms such as in ICMPv6 [11]. Another possibility would be to use ECN [17], if the datagram mode messages can be correlated with a congestion controlled messaging association which also supports ECN. Details are clearly for further study.

8.11 Message Format Issues

NSIS message formats are defined as a set of objects (see Appendix C.1). Some aspects are left open:

Ordering: Traditionally, Internet protocols require a node to be able to process a message with objects in any order. However, this has some costs in parser complexity, testing interoperability, ease of compression; there is a special issue with GIMPS that for efficiency, the NTLP-Data object (which may be large) should come

last. Should object order be fixed or unspecified?

Capabilities: For extensibility, it is useful to be able to mark objects with some information about how they should be treated if the receiving node does not implement them (e.g. ignore or reject). Since the object space is shared between all protocols, this marking has to be standardised across all the NSIS protocols. Is an object marking scheme based on some flags in the object header appropriate, or a more flexible scheme based on some type of capability encoding?

8.12 Protocol Design Details

Clearly, not all details of GIMPS operation have been defined so far. This section provides a list of slightly non-trivial areas where more detail is need, where these have not been mentioned elsewhere in the text.

- Receiver initiated signaling applications need to have reverse path state set up in the network, before the signaling application itself can originate any messages. Should this be done by GIMPS carrying out the discovery for the specific signaling application (which requires the flow sender to know what signaling applications are going to be used), or should the discovery attempt to find every GIMPS node and the signaling applications they support?

9. Change History

9.1 Changes In Version -02

Version -02 does not represent any radical change in design or structure from version -01; the emphasis has been on adding details in some specific areas and incorporation of comments, including early review comments. The full list of changes is as follows:

1. Added a new Section 1.1 which summarises restrictions on scope and applicability; some corresponding changes in terminology in Section 2.

2. Closed the open issue on including explicit GIMPS state teardown functionality. On balance, it seems that the difficulty of specifying this correctly (especially taking account of the security issues in all scenarios) is not matched by the saving of state enabled.

3. Removed the option of a special class of message transfer for reliable delivery of a single message. This can be implemented (inefficiently) as a degenerate case of C-mode if required.

4. Extended Appendix C with a general discussion of rules for message and object formats across GIMPS and other NSLPs. Some remaining open issues are noted in Section 8.11.

5. Updated the discussion of Section 8.4 to take into account the proposed message formats and rules for allocation of NSLP id, and propose considerations for allocation of RAO values.

6. Modified the description of the information used to route messages (first given in Section 4.1.1 but also throughout the document). Previously this was related directly to the flow identification and described as the Flow-Routing-Information. Now, this has been renamed Message-Routing-Information, and identifies a message routing method and any associated addressing.

7. Modified the text in Section 4.2 and elsewhere to impose sanity checks on the Message-Routing-Information carried in C-mode messages, including the case where these messages are part of a GIMPS-Query/Response exchange.

8. Added rules for message forwarding to prevent message looping in a new Section 4.2.4, including rules on IP TTL and GIMPS hop count processing. These take into account the new RAO considerations of Section 8.4.

9. Added an outline mechanism for messaging association protocol stack negotiation, with the details in a new Section 6.6 and other changes in Section 4.3 and the various sections on message formats.

10. Removed the open issue on whether storing reverse routing state is mandatory or optional. This is now explicit in the API (under the control of the local NSLP).

11. Added an informative annex describing an abstract API between GIMPS and NSLPs in Appendix D.

9.2 Changes In Version -01

The major change in version -01 is the elimination of 'intermediaries', i.e. imposing the constraint that signaling application peers are also GIMPS peers. This has the consequence that if a signaling application wishes to use two classes of signaling transport for a given flow, maybe reaching different subsets of nodes, it must do so by running different signaling sessions; and it also means that signaling adaptations for passing through NATs which are not signaling application aware must be carried out in datagram mode. On the other hand, it allows the elimination of significant complexity in the connection mode handling and also various other protocol features (such as general route recording).

The full set of changes is as follows:

1. Added a worked example in Section 3.3.

2. Stated that nodes which do not implement the signaling application should bypass the message (Section 4.2).

3. Decoupled the state handling logic for routing state and messaging association state in Section 4.3. Also, allow messaging associations to be used immediately in both directions once they are opened.

4. Added simple ABNF for the various GIMPS message types in a new Section 5.1, and more details of the common header and each object in Section 5.2, including bit formats in Appendix C. The common header format means that the encapsulation is now the same for all transport types (Section 5.4.1).

5. Added some further details on datagram mode encapsulation in Section 5.3, including more explanation of why a well known port

it needed.

6. Removed the possibility for fragmentation over DCCP (Section 5.4.1), mainly in the interests of simplicity and loss amplification.

7. Removed all the tunnel mode encapsulations (old sections 5.3.3 and 5.3.4).

8. Fully re-wrote the route change handling description (Section 6.1), including some additional detection mechanisms and more clearly distinguishing between upstream and downstream route changes. Included further details on GIMPS/NSLP interactions, including where notifications are delivered and how local repair storms could be avoided. Removed old discussion of propagating notifications through signaling application unaware nodes (since these are now bypassed automatically). Added discussion on how to route messages for local state removal on the old path.

9. Revised discussion of policy-based forwarding (Section 6.2) to account for actual FLow-Routing-Information definition, and also how wildcarding should be allowed and handled.

10. Removed old route recording section (old Section 6.3).

11. Extended the discussion of NAT handling (Section 6.3) with an extended outline on processing rules at a GIMPS-aware NAT and a pointer to implications for C-mode processing and state management.

12. Clarified the definition of 'correct routing' of signaling messages in Section 7 and GIMPS role in enforcing this. Also, opened the possibility that peer node authentication could be signaling application dependent.

13. Removed old open issues on Connection Mode Encapsulation (section 8.7); added new open issues on Message Routing (Section 8.9) and Datagram Mode congestion control (Section 8.10).

14. Added this change history.

10. References

10.1 Normative References

[1] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

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

[3] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[4] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, October 1999.

[5] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[6] 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.

[7] Kohler, E., "Datagram Congestion Control Protocol (DCCP)", draft-ietf-dccp-spec-06 (work in progress), February 2004.

[8] Stewart, R., "SCTP Partial Reliability Extension", draft-ietf-tsvwg-prsctp-03 (work in progress), January 2004.

10.2 Informative References

[9] Braden, B., Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[10] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[11] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

[12] Terzis, A., Krawczyk, J., Wroclawski, J. and L. Zhang, "RSVP Operation Over IP Tunnels", RFC 2746, January 2000.

[13] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[14] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F. and S.

Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

[15] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[16] Huitema, C., "An Anycast Prefix for 6to4 Relay Routers", RFC 3068, June 2001.

[17] Ramakrishnan, K., Floyd, S. and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[18] Baker, F., Iturralde, C., Le Faucheur, F. and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.

[19] 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.

[20] Arkko, J., Torvinen, V., Camarillo, G., Niemi, A. and T. Haukka, "Security Mechanism Agreement for the Session Initiation Protocol (SIP)", RFC 3329, January 2003.

[21] Rosenberg, J., Weinberger, J., Huitema, C. and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[22] Hancock, R., "Next Steps in Signaling: Framework", draft-ietf-nsis-fw-05 (work in progress), October 2003.

[23] Tschofenig, H. and D. Kroeselberg, "Security Threats for NSIS", draft-ietf-nsis-threats-04 (work in progress), February 2004.

[24] Stiemerling, M., "A NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", draft-ietf-nsis-nslp-natfw-02 (work in progress), May 2004.

[25] Bosch, S., Karagiannis, G. and A. McDonald, "NSLP for Quality-of-Service signaling", draft-ietf-nsis-qos-nslp-03 (work in progress), May 2004.

[26] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", draft-ietf-v6ops-mech-v2-02 (work in progress), February 2004.

[27] Nikander, P., "Mobile IP version 6 Route Optimization Security Design Background", draft-nikander-mobileip-v6-ro-sec-02 (work in progress), December 2003.

[28] Bound, J., "Dual Stack Transition Mechanism", draft-bound-dstm-exp-01 (work in progress), April 2004.

Authors' Addresses

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

Robert Hancock
Siemens/Roke Manor Research

Old Salisbury Lane
Romsey, Hampshire SO51 0ZN
UK

EMail: robert.hancock@roke.co.uk
URI: http://www.roke.co.uk

Appendix A. Acknowledgements

This document is based on the discussions within the IETF NSIS working group. It has been informed by prior work and formal and informal inputs from: Cedric Aoun, Attila Bader, Bob Braden, Marcus Brunner, Xiaoming Fu, Ruediger Geib, Eleanor Hepworth, Georgios Karagiannis, John Loughney, Jukka Manner, Andrew McDonald, Glenn Morrow, Dave Oran, Charles Shen, Melinda Shore, Martin Stiemerling, Mike Thomas, Hannes Tschofenig, Sven van den Bosch, Michael Welzl, and Lars Westberg. In particular, Hannes Tschofenig provided a detailed set of review comments on the security section, and Andrew McDonald provided the formal description for the initial packet

formats. We look forward to inputs and comments from many more in the future.

Appendix B. Example Message Routing State Table

Figure 7 shows a signaling scenario for a single flow being managed by two signaling applications. The flow sender and receiver and one router support both, two other routers support one each.

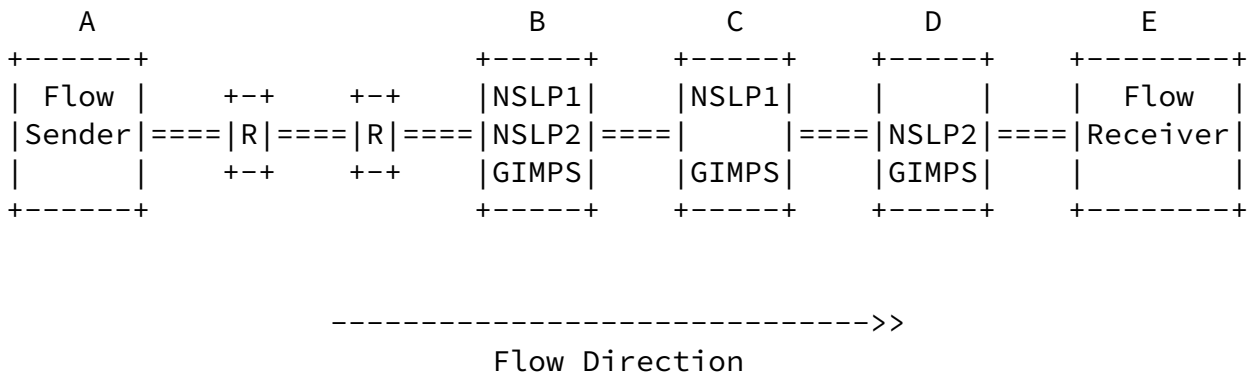


Figure 7: A Signaling Scenario

Routing state table at node B:

Message Routing Information	Session ID	NSLP ID	Upstream Peer	Downstream Peer
Method = Path Coupled; Flow ID = {IP-#A, IP-#E, protocol, ports}	0xABCD	NSLP1	IP-#A	(null)
Method = Path Coupled; Flow ID = {IP-#A, IP-#E, protocol, ports}	0x1234	NSLP2	IP-#A	Pointer to B-D messaging association

The table shows the routing state at Node B for the single flow from A to E. The upstream state is just the same address for each application. For the downstream case, NSLP1 only requires datagram mode messages and so no explicit routing state towards C is needed. NSLP2 requires a messaging association for its messages towards node D, and node C does not process NSLP2 at all, so the downstream peer state for NSLP2 is a pointer to a messaging association that runs directly from B to D. Note that E is not visible in the state table (except implicitly in the address in the message routing information); routing state is stored only for adjacent peers.

Appendix C. Bit-Level Formats

This appendix provides initial formats for the various component parts of the GIMPS messages defined abstractly in Section 5.2. It should be noted that these formats are extremely preliminary and should be expected to change completely several times during the further development of this specification.

In addition, this appendix includes some general rules for the format of messages and message objects across all protocols in the NSIS protocol suite (i.e. the current and future NSLPs as well as GIMPS itself). The intention of these common rules is to encourage commonality in implementations, ease of testing and debugging, and sharing of object definitions across different applications.

C.1 General NSIS Formatting Guidelines

Each NSIS message consists of a header and a sequence of objects. An NSLP message is one object within a GIMPS message. The GIMPS header has a specific format, described in more detail in Appendix C.2 below; the NSLP header format is common to all signaling applications and includes simply a message type (which may be structured into a type field and some processing flags, depending on the application). Note that GIMPS provides the message length information and signaling application identification. There is no version information; if an NSLP is extended so much that it stops being backwards compatible, a new signaling application identifier is allocated.

Every object has the following general format:

- The overall format is Type-Length-Value (in that order).

- Assignments for the Type field are common across all NSIS protocols (i.e. there is a single registry). This is to facilitate the sharing of common objects across different signaling applications. How to encode capability information therefore has to be standardised across signaling applications; this is still an open issue (see Section 8.11.

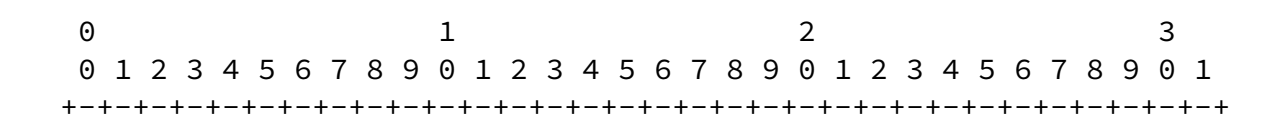
- Length has the units of 32 bit words, and measures the length of Value. If there is no Value, Length=0.

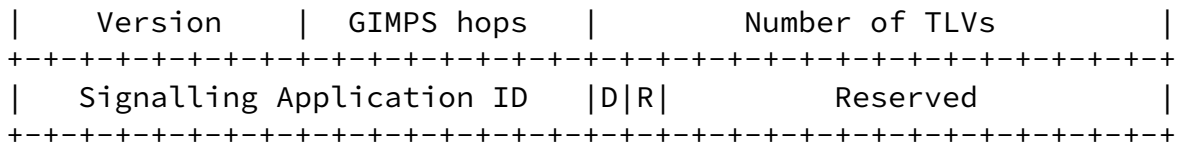
- Value is (therefore) a whole number of 32 bit words. If there is any padding required, this must be defined by the object-specific format information; objects which contain variable length (e.g. string) types may need to include additional length subfields to do so.

Error messages are identified by containing an error object (i.e. an object with Type='Error'). There should be a common error object format, whose Value field includes a severity indication, an error code, and optionally additional error-specific information. Again, the error code space is common across all protocols.

C.2 The GIMPS Common Header

This header precedes all GIMPS messages. It has a fixed format, as shown below. Note that (unlike NSLP messages) the GIMPS header does include a version number, since allocating new lower layer identifiers to demultiplex a new GIMPS version will be significantly harder than allocating a new NSLP identifier.



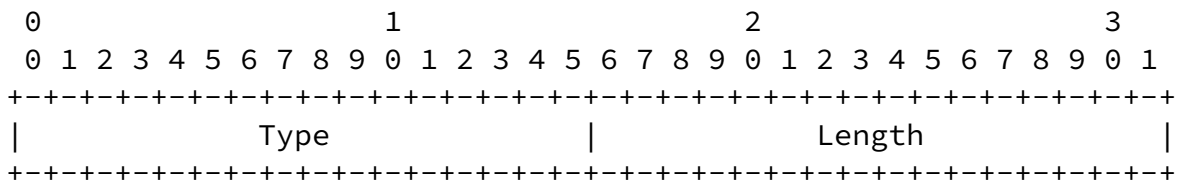


The flags are:
D - Direction (Set for "Upstream", Unset for "Downstream")
R - Response requested

C.3 GIMPS TLV Objects

C.3.1 Standard Object Header

Each object begins with a fixed header giving the object type and object length. See Section 8.11 for a discussion of extensibility issues for object encoding.

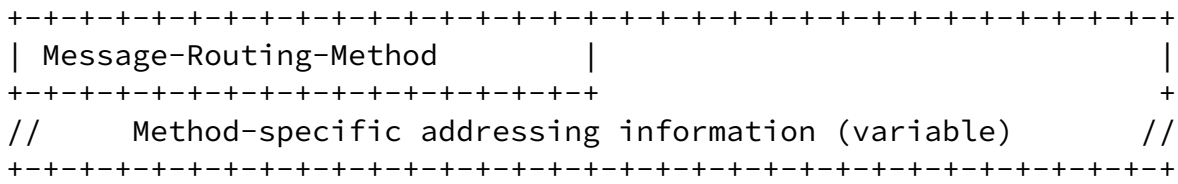


In the following object diagrams, '//' is used to indicate a variable sized field and ':' is used to indicate a field that is optionally present.

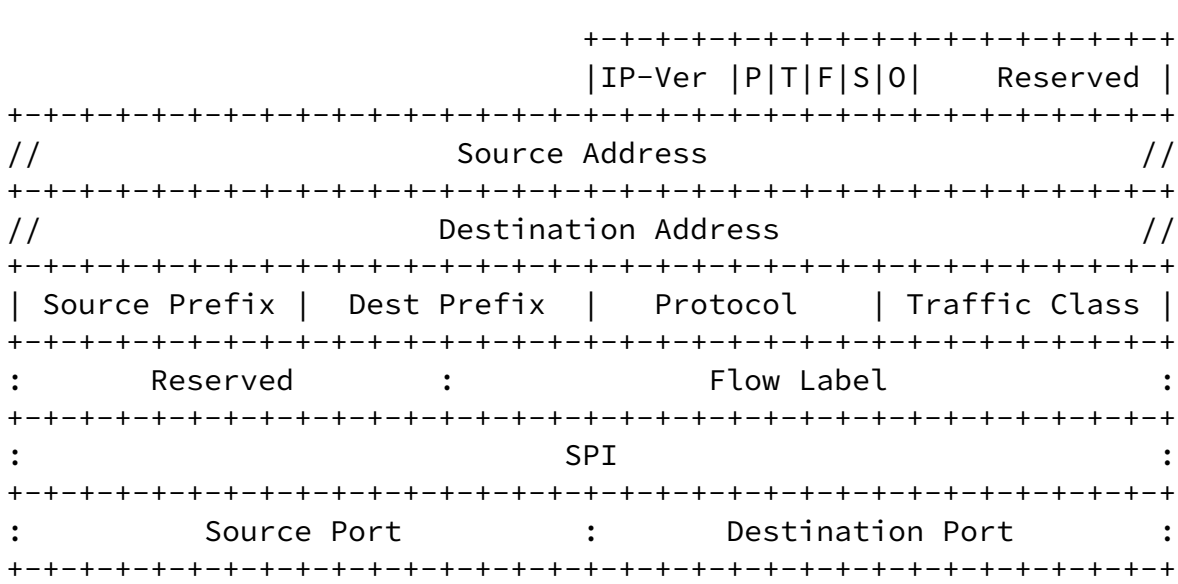
C.3.2 Message-Routing-Information

Type: Message-Routing-Information

Length: Variable (depends on message routing method)



In the case of basic path-coupled routing, the addressing information takes the following format:

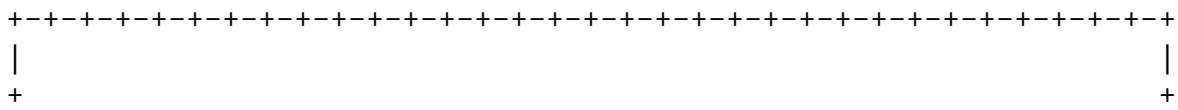


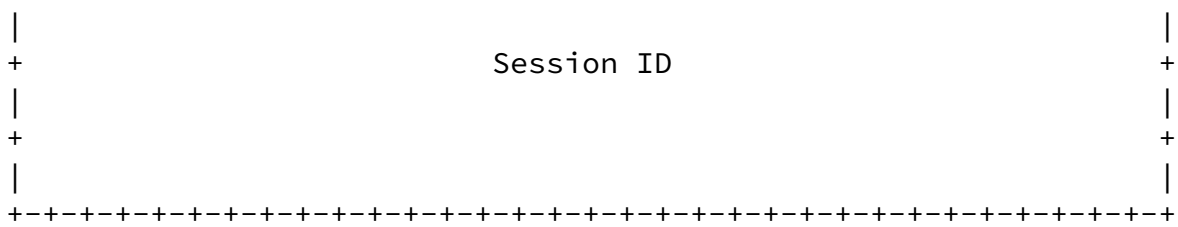
The flags are:
P - Protocol
T - Traffic Class
F - Flow Label
S - SPI
O - Source/Destination Ports

C.3.3 Session Identification

Type: Session-Identification

Length: Fixed (TBD 4 32-bit words)

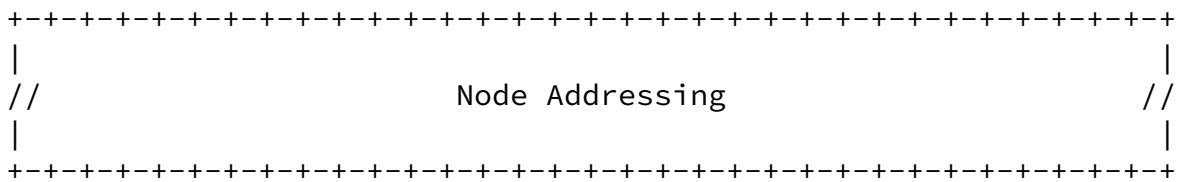




C.3.4 Node Addressing

Type: Node-Addressing

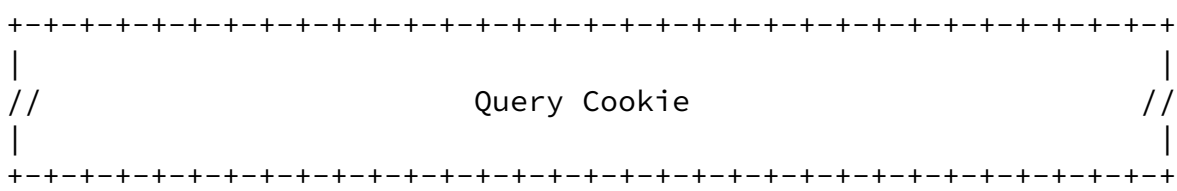
Length: Variable (depends on detailed format and what optional fields are present)



C.3.5 Query Cookie

Type: Query-Cookie

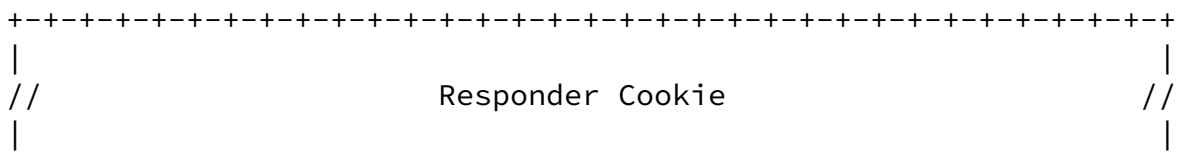
Length: Variable (selected by querying node)



C.3.6 Responder Cookie

Type: Responder-Cookie

Length: Variable (selected by querying node)

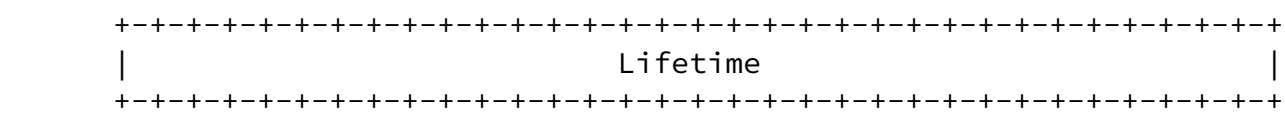


C.3.7 Lifetime

Type: Lifetime

Length: Fixed - 1 32-bit word

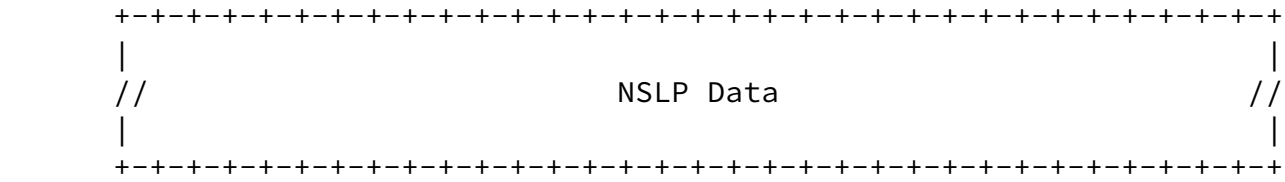
Value: Routing state lifetime in seconds



C.3.8 NSLP Data

Type: NSLP-Data

Length: Variable (depends on NSLP)



Appendix D. API between GIMPS and NSLP

This appendix provides an initial abstract API between GIMPS and NSLPs.

This does not constrain implementors, but rather helps clarify the interface between the different layers of the NSIS protocol suite. In addition, although some of the data types carry the information from GIMPS Information Elements, this does not imply that the format of that data as sent over the API is the same.

Conceptually the API has similarities to the UDP sockets API, particularly that for unconnected UDP sockets. An extension for an API like that for UDP connected sockets could be considered. In this case, for example, the only information needed in a SendMessage primitive would be NSLP-Data, NSLP-Data-Size, and NSLP-Message-Handle (which can be null). Other information which was persistent for a group of messages could be configured once for the socket.

D.1 SendMessage

This primitive is passed from an NSLP to GIMPS. It is used whenever the NSLP wants to send a message.

```
SendMessage ( NSLP-Data, NSLP-Data-Size, NSLP-Message-Handle,
              NSLP-Id, Session-ID,
              MRI, Direction, SII-Handle,
              Transfer-Attributes, Timeout, IP-TTL )
```

NSLP-Data: The NSLP message itself.

NSLP-Data-Size: The length of NSLP-Data.

NSLP-Message-Handle: A handle for this message, that can be used later by GIMPS to reference it in error messages, etc. A NULL handle may be supplied if the NSLP is not interested in receiving MessageDeliveryError notifications for this message.

NSLP-Id: An identifier indicating which NSLP this is.

Session-ID: The NSIS session identifier.

MRI: Message routing information for use by GIMPS in determining the correct next GIMPS hop for this message. It contains, for example, the flow source/destination addresses and the type of routing to use for the signalling message.

Direction: A flag indicating whether the message is to be sent in the upstream or downstream direction (in relation to the MRI).

SII-Handle: A handle, previously supplied by GIMPS, to a data structure (identifying peer addresses and interfaces) that should be used to explicitly route the message to a particular GIMPS next hop. If supplied, GIMPS should validate that it is consistent with the MRI.

Transfer-Attributes: Reliability, security, priority etc. attributes to be used for sending this particular message. A value indicating "default" or "don't care" may be given.

Timeout: Length of time GIMPS should attempt to send this message before indicating an error. A value indicating "default" or "don't care" may be given.

IP-TTL: The value of the IP TTL that should be used when sending this message. A value indicating "default" or "don't care" may be given.

D.2 RecvMessage

This primtive is passed from GIMPS to an NSLP. It is used whenever GIMPS receives a message.

```
RecvMessage ( [NSLP-Data, NSLP-Data-Size,]
              NSLP-Id, Session-ID,
              MRI, Direction, SII-Handle,
              Transfer-Attributes,
              IP-TTL, Original-TTL )
```

NSLP-Data: The NSLP message itself (may be empty).

NSLP-Data-Size: The length of NSLP-Data (may be zero).

NTLP-Message-Handle: A handle for this message, that can be used later by the NSLP to reference it in a MessageReceived primitive.

NSLP-Id: An identifier indicating which NSLP this is message is for.

Session-ID: The NSIS session identifier.

MRI: Message routing information that was used by GIMPS in forwarding this message. It contains, for example, the flow source/ destination addresses and the type of routing to used for the signalling message.

Direction: A flag indicating whether the message was received going in the upstream or downstream direction (in relation to the MRI).

SII-Handle: A handle to a data structure, identifying peer addresses and interfaces. Can be used to identify route changes and for explicit routing to a particular GIMPS next hop.

Transfer-Attributes: Reliability, security, priority, etc. attributes that were used for this particular message.

IP-TTL: The value of the IP TTL (or Hop Limit) associated with this message.

Original-TTL: The value of the IP TTL (or Hop Limit) at the time of sending of the packet that contained this message.

D.3 MessageReceived

This primitive is passed from an NSLP to GIMPS. It is used after a RecvMessage primitive has been passed from GIMPS to an NSLP to inform GIMPS whether the NSLP wishes GIMPS to retain state.

MessageReceived (NTLP-Message-Handle, Retain-State)

NTLP-Message-Handle: A handle on a message, previously supplied by GIMPS in a RecvMessage primitive.

RetainState: A value indicating whether or not the NSLP wishes GIMPS to retain path state.

D.4 MessageDeliveryError

This primitive is passed from GIMPS to an NSLP. It is used to notify the NSLP that a message that it requested to be sent has failed to be dispatched.

MessageDeliveryError (NSLP-Message-Handle, Error-Type)

NSLP-Message-Handle: A handle for the message provided by the NSLP at

the time of sending.

Error-Type: Indicates the type of error that occurred. For example, 'no next node found'.

D.5 NetworkNotification

This primitive is passed from GIMPS to an NSLP. It indicates that a network event of possible interest to the NSLP occurred.

NetworkNotification (MRI, Network-Notification-Type)

MRI: Provides the message routing information to which the network notification applies.

Network-Notification-Type: Indicates the type of event that caused the notification, e.g. downstream route change, upstream route change, detection that this is the last node.

D.6 SecurityProtocolAttributesRequest

This primitive is passed from GIMPS to an NSLP. It is sent when GIMPS requires the NSLP to make decisions (e.g. check policy) or provide information for authentication parameters to be used when setting up a messaging association.

SecurityProtocolAttributesRequest (Peer-Info, Security-Protocol, Request-Ty

Peer-Info: Information identifying the GIMPS peer and interface

Security-Protocol: A value indicating the security protocol being used (TLS, IPsec, etc).

Request-Type: An indication of the type of information required (e.g. client certificate)

D.7 SetStateLifetime

This primitive is passed from an NSLP to GIMPS. It indicates the lifetime for which the NSLP would like GIMPS to retain its state. It can also give a hint that the NSLP is no longer interested in the state.

SetStateLifetime (MRI, Direction, State-Lifetime)

MRI: Provides the message routing information to which the network notification applies.

Direction: A flag indicating whether this relates to state for the upstream or downstream direction (in relation to the MRI).

State-Lifetime: Indicates the lifetime for which the NSLP wishes GIMPS to retain its state (may be zero, indicating that the NSLP has no further interest in the GIMPS state).

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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