INTERNET-DRAFT
Expires: December 1999
<draft-ietf-rsvp-diagnostic-msgs-08.txt>

Andreas Terzis
UCLA
Bob Braden
ISI
Subramaniam Vincent
Cisco Systems
Lixia Zhang
UCLA

June 1999

RSVP Diagnostic Messages

<draft-ietf-rsvp-diagnostic-msgs-08.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt

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

Abstract

This document specifies the RSVP diagnostic facility, which allows a user to collect information about the RSVP state along a path. This specification describes the functionality, diagnostic message formats, and processing rules.

Changes

A summary of the changes from the previous version (07) of this document follows:

The processing rules have been rewritten in a more compact form.

1. Introduction

In the basic RSVP protocol [RSVP], error messages are the only means for an end host to receive feedback regarding a failure in setting up either path state or reservation state. An error message carries back only the information from the failed point, without any information about the state at other hops before or after the failure. In the absence of failures, a host receives no feedback regarding the details of a reservation that has been put in place, such as whether, or where, or how, its own reservation request is being merged with that of others. Such missing information can be highly desirable for debugging purposes, or for network resource management in general.

This document specifies the RSVP diagnostic facility, which is designed to fill this information gap. The diagnostic facility can be used to collect and report RSVP state information along the path from a receiver to a specific sender. It uses Diagnostic messages that are independent of other RSVP control messages and produce no side-effects; that is, they do not change any RSVP state at either nodes or hosts. Similarly, they provide not an error report but rather a collection of requested RSVP state information.

The RSVP diagnostic facility was designed with the following goals:

- To collect RSVP state information from every RSVP-capable hop along a path defined by path state, either for an existing reservation or before a reservation request is made. More specifically, we want to be able to collect information about flowspecs, refresh timer values, and reservation merging at each hop along the path.

- To collect the IP hop count across each non-RSVP cloud.

- To avoid diagnostic packet implosion or explosion.

The following is specifically identified as a non-goal:

- Checking the resource availability along a path. Such functionality may be useful for future reservation requests, but it would require modifications to existing admission control modules that is beyond the scope of RSVP.

2. Overview

The diagnostic facility introduces two new RSVP message types: Diagnostic Request (DREQ) and Diagnostic Reply (DREP). A DREQ message can be originated by a client in a "requester" host, which may or may not be a participant of the RSVP session to be diagnosed. A client in the requester host invokes the RSVP diagnostic facility by generating a DREQ packet and sending it towards the LAST-HOP node, which should be on the RSVP path to be diagnosed. This DREQ packet specifies the RSVP session and a sender host for that session. Starting from the LAST-HOP, the DREQ packet collects information hop-by-hop as it is forwarded towards the sender (see Figure 1), until it reaches the ending node. Specifically, each RSVP-capable hop adds to the DREQ message a response (DIAG_RESPONSE) object containing local RSVP state for the specified RSVP session.

When the DREQ packet reaches the ending node, the message type is changed to Diagnostic Reply (DREP) and the completed response is sent to the original requester node. Partial responses may also be returned before the DREQ packet reaches the ending node if an error condition along the path, such as "no path state", prevents further forwarding of the DREQ packet. To avoid packet implosion or explosion, all diagnostic packets are forwarded via unicast only.

Thus, there are generally three nodes (hosts and/or routers) involved in performing the diagnostic function: the requester node, the starting node, and the ending node, as shown in Figure 1. It is possible that the client invoking the diagnosis function may reside directly on the starting node, in which case that the first two nodes are the same. The starting node is named "LAST-HOP", meaning the last-hop of the path segment to be diagnosed. The LAST-HOP node can be either a receiver node or an intermediate node along the path. The ending node is usually the specified sender host. However, the client can limit the length of the path segment to be diagnosed by specifying a hop-count limit in the DREQ message.

```
                    LAST-HOP                  Ending
       Receiver        node                     node           Sender
           __           __         __            __              __
          |  |---------|  |------>|  |--> ...-->|  |--> ...---->|  |
          |__|         |__| DREQ  |__|   DREQ   |__|   DREQ     |__|
                        ^                         .              |
                        |                         .              |
                        | DREQ                    . DREP         | DREP
                        |                         .              |
                       _|_               DREP     V              V
          Requester   |   | <-------------------------------------
          (client)    |___|
```

Figure 1

DREP packets can be unicast from the ending node back to the requester either directly or hop-by-hop along the reverse of the path taken by the DREQ message to the LAST-HOP, and thence to the requester. The direct return is faster and more efficient, but the hop-by-hop reverse-path route may be the only choice if the packets have to cross firewalls. Hop-by-hop return is accomplished using an optional ROUTE object, which is built incrementally to contain a list of node addresses that the DREQ packet has passed through. The ROUTE object is then used in reverse as a source route to forward the DREP hop-by-hop back to the LAST-HOP node.

A DREQ message always consists of a single unfragmented IP datagram. On the other hand, one DREQ message can generate multiple DREP packets, each containing a fragment of the total DREQ message. When the path consists of many hops, the total length of a DREP message will exceed the MTU size before reaching the ending node; thus, the message has to be fragmented. Relying on IP fragmentation and reassembly, however, can be problematic, especially when DREP messages are returned to the requester hop-by-hop, in which case fragmentation/reassembly would have to be performed at every hop. To avoid such excessive overhead, we let the requester define a default path MTU size that is carried in every DREQ packet. If an intermediate node finds that the default MTU size is bigger than the MTU of the incoming interface, it reduces the default MTU size to the MTU size of the incoming interface. If an intermediate node detects that a DREQ packet size is larger than the default MTU size, it returns to the requester (in either manner described above) a DREP fragment containing accumulated responses. It then removes these responses from the DREQ and continues to forward it. The requester node can reassemble the resulting DREP fragments into a complete DREP message.

When discussing diagnostic packet handling, this document uses direction terminology that is consistent with the RSVP functional specification [RSVP], relative to the direction of data packet flow. Thus, a DREQ packet enters a node through an "outgoing interface" and is forwarded towards the sender through an "incoming interface", because DREQ packets travel in the reverse direction to the data flow.

Notice that DREQ packets can be forwarded only after the RSVP path state has been set up. If no path state exists, one may resort to the traceroute or mtrace facility to examine whether the unicast/multicast routing is working correctly.

3. Diagnostic Packet Format

Like other RSVP messages, DREQ and DREP messages consist of an RSVP Common Header followed by a variable set of typed RSVP data objects. The following sequence must be used:

```
+-----------------------------------+
|        RSVP Common Header         |
+-----------------------------------+
|         Session object            |
+-----------------------------------+
|      Next-Hop RSVP_HOP object     |
+-----------------------------------+
|       DIAGNOSTIC object           |
+-----------------------------------+
|    (optional) DIAG_SELECT object  |
+-----------------------------------+
|    (optional) ROUTE object        |
+-----------------------------------+
| zero or more DIAG_RESPONSE objects|
+-----------------------------------+
```

The session object identifies the RSVP session for which the state information is being collected. We describe each of the other parts.

3.1. RSVP Message Common Header

The RSVP message common header is defined in [RSVP]. The following specific exceptions and extensions are needed for DREP and DREQ.

Type field: define:

Type = 8: DREQ Diagnostic Request

Type = 9: DREP Diagnostic Reply

RSVP length:

If this is a DREP message and the MF flag in the DIAGNOSTIC object (see below) is set, this field indicates the length of this single DREP fragment rather than the total length of the complete DREP reply message (which cannot generally be known in advance).

3.2. Next-Hop RSVP_HOP Object

This RSVP_HOP object carries the LIH of the interface through which the DREQ should be received at the upstream node. This object is updated hop-by hop. It is used for the same reasons that a RESV message contains an RSVP_HOP object: to distinguish logical interfaces and avoid problems caused by routing asymmetries and non-RSVP clouds.

While the IP address is not really used during DREQ processing , for consistency with the use of the RSVP_HOP object in other RSVP messages, the IP address in the RSVP_HOP object to contain the address of the interface through which the DREQ was sent.

3.3. DIAGNOSTIC Object

A DIAGNOSTIC object contains the common diagnostic control information in both DREQ and DREP messages.

o IPv4 DIAGNOSTIC object: Class = 30, C-Type = 1

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Max-RSVP-hops | RSVP-hop-count|         Reserved            |MF|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          Request ID                             |
 +---------------+---------------+---------------+---------------+
 |           Path MTU            |     Fragment Offset           |
 +---------------+---------------+---------------+---------------+
 |                         LAST-HOP Address                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                                |
 |                     SENDER_TEMPLATE object                     |
 |                                                                |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                                |
 |                 Requester FILTER_SPEC object                   |
 |                                                                |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Here all IP addresses use the 4 byte IPv4 format, both explicitly in the LAST-HOP Address and by using the IPv4 forms of the embedded FILTER_SPEC and RSVP_HOP objects.

o IPv6 DIAGNOSTIC object: Class = 30, C-Type = 2

The format is the same, except all explicit and embedded IP addresses are 16 byte IPv6 addresses.

The fields are as follows:

Max-RSVP-hops

An octet specifying the maximum number of RSVP hops over which information will be collected. If an error condition in the middle of the path prevents the DREQ packet from reaching the specified ending node, the Max-RSVP-hops field may be used to perform an expanding-length search to reach the point just before the problem. If this value is 1, the starting node and the ending node of the query will be the same. If it is zero, there is no hop limit.

RSVP-hop-count

Records the number of RSVP hops that have been traversed so far. If the starting and ending nodes are the same, this value will be 1 in the resulting DREP message.

Fragment Offset

Indicates where this DREP fragment belongs in the complete DREP message, measured in octets. The first fragment has offset zero. Fragment Offset is used also to determine if a DREQ message containing zero DIAG_RESPONSE objects should be processed at an RSVP capable node.

MF flag

Flag means "more fragments". It must be set to zero (0) in all DREQ messages. It must be set to one (1) in all DREP packets that carry partial results and are returned by intermediate nodes due to the MTU limit. When the DREQ message is converted to a DREP message in the ending node, the MF flag must remain zero.

Request ID

Identifies an individual DREQ message and the corresponding DREP message (or all the fragments of the reply message).

One possible way to define the Request ID would use 16 bits to specify the ID of the process making the query and 16 bits to distinguish different queries from this process.

Path MTU

Specifies a default MTU size in octets for DREP and DREQ messages. This value should not be smaller than the size of the "base" DREQ packet. A "base" DREQ packet is one that contains a Common Header, a Session object, a Next-Hop RSVP_HOP object, a DIAGNOSTIC object, an empty ROUTE object and a single default DIAG_RESPONSE (see below). The assumption made here is that a diagnostic packet of this size can always be forwarded without IP fragmentation.

LAST-HOP Address

The IP address of the LAST-HOP node. The DREQ message starts collecting information at this node and proceeds toward the sender.

SENDER_TEMPLATE object

This IPv4/IPv6 SENDER_TEMPLATE object contains the IP address and the port of a sender for the session being diagnosed. The DREQ packet is forwarded hop-by-hop towards this address.

Requester FILTER_SPEC Object

This IPv4/IPv6 FILTER_SPEC object contains the IP address and the port from which the request originated and to which the DREP message(s) should be sent.

3.4. DIAG_SELECT Object

o DIAG_SELECT Class = 33, C-Type = 1.

A Diagnostic message may optionally contain a DIAG_SELECT object to specify which specific RSVP objects should be reported in a DIAG_RESPONSE object. In the absence of a DIAG_SELECT object, the DIAG_RESPONSE object added by the node will contain a default set of object types (see DIAG_RESPONSE object below).

The DIAG_SELECT object contains a list of [Class, C-type] pairs, in the following format:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    class      |     C-Type    |    class      |     C-Type    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 //                                                              //
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    class      |     C-Type    |    class      |     C-Type    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

When a DIAG_SELECT object is included in a DREQ message, each RSVP node along the path will add a DIAG_RESPONSE object containing response objects (see below) whose classes and C-Types match entries in the DIAG_SELECT list (and are from matching path and reservation state). A C-type octet of zero is a 'wildcard', matching any C-Type associated with the associated class.

Depending on the type of objects requested, a node can find the associated information in the path or reservation state stored for the session described in the SESSION object. Specifically, information for the RSVP_HOP,SENDER_TEMPLATE, SENDER_TSPEC, ADSPEC objects can be extracted from the node's path state, while information for the FLOWSPEC, FILTER_SPEC, CONFIRM, STYLE and SCOPE objects can be found in the node's reservation state (if existent).

If the number of [Class, C-Type] pairs is odd, the last two octets of the DIAG_SELECT object must be zero. A maximum DIAG_SELECT object is

one that contains the [Class, C-type] pairs for all the RSVP objects that can be requested in a Diagnostic query.

3.5. ROUTE Object

A diagnostic message may contain a ROUTE object, which is used to record the route of the DREQ message and as a source route for returning the DREP message(s) hop-by-hop.

o IPv4 ROUTE object: Class = 31, C-Type = 1.

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             reserved                          |    R-pointer  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                     RSVP Node List                            |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This message signifies how the reply should be returned. If it does not exist in the DREQ packet then DREP packets should be sent to the requester directly. If it does exist, DREP packets must be returned hop-by-hop along the reverse path to the LAST-HOP node and thence to the requester node.

An empty ROUTE object is one that has an empty RSVP Node list and R-pointer is equal to zero.

RSVP Node List

A list of RSVP node IPv4 addresses. The number of addresses in this list can be computed from the object size.

R-pointer

Used in DREP messages only (see Section 4.2 for details), but it is incremented as each hop adds its incoming interface address in the ROUTE object.

o IPv6 ROUTE object: Class = 31, C-Type = 2

The same, except RSVP Node List contains IPv6 addresses.

In a DREQ message, RSVP Node List specifies all RSVP hops between the

LAST-HOP address specified in the DIAGNOSTIC object, and the last RSVP node the DREQ message has visited. In a DREP message, RSVP Node List specifies all RSVP hops between the LAST-HOP and the node that returns this DREP message.

3.6. DIAG_RESPONSE Object

Each RSVP node attaches a DIAG_RESPONSE object to each DREQ message it receives, before forwarding the message. The DIAG_RESPONSE object contains the state to be reported for this node. It has a fixed-format header and then a variable list of RSVP state objects, or "response objects".

o IPv4 DIAG_RESPONSE object: Class = 32, C-Type = 1.

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       DREQ Arrival Time                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Incoming Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Outgoing Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Previous-RSVP-Hop Router Address              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   D-TTL       |M|R-err|  K    |      Timer value              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 |                  (optional) TUNNEL object                     |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 //                       Response objects                      //
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

o IPv6 DIAG_RESPONSE object: Class = 32, C-Type = 2.

This object has the same format, except that all explicit and embedded IP addresses are IPv6 addresses.

The fields are as follows:

DREQ Arrival Time

A 32-bit NTP timestamp specifying the time the DREQ message arrived at this node. The 32-bit form of an NTP timestamp consists of the middle 32 bits of the full 64-bit form, that is, the low 16 bits of the integer part and the high 16 bits of the fractional part.

Incoming Interface Address

Specifies the IP address of the interface on which messages from the sender are expected to arrive, or 0 if unknown.

Outgoing Interface Address

Specifies the IP address of the interface through which the DREQ message arrived and to which messages from the given sender and for the specified session address flow, or 0 if unknown.

Previous-RSVP-Hop Router Address

Specifies the IP address from which this node receives RSVP PATH messages for this source, or 0 if unknown. This is also the interface to which the DREQ will be forwarded.

D-TTL

The number of IP hops this DREQ message traveled from the down-stream RSVP node to the current node.

M flag

A single-bit flag which indicates whether the reservation described by the response objects is merged with reservations from other down-stream interfaces when being forwarded upstream.

R-error

A 3-bit field that indicates error conditions at a node. Currently defined values are:

```
0x00: no error
0x01: No PATH state
0x02: packet too big
0x04: ROUTE object too big
```

K

The refresh timer multiple (defined in [RSVP]).

Timer value

The local refresh timer value in seconds.

The set of response objects to be included at the end of the DIAG_RESPONSE object is determined by a DIAG_SELECT object, if one is present. If no DIAG_SELECT object is present, the response objects belong to the default list of classes:

SENDER_TSPEC object FILTER_SPEC object FLOWSPEC object
STYLE object

Any C-Type present in the local RSVP state will be used. These response objects may be in any order but they must all be at the end of the DIAG_RESPONSE object.

A default DIAG_RESPONSE object is one containing the default list of classes described above.

3.7. TUNNEL Object

The optional TUNNEL object should be inserted when a DREQ message arrives at an RSVP node that acts as a tunnel exit point.

The TUNNEL object provides the mapping between the end-to-end RSVP session that is being diagnosed and the RSVP session over the tunnel. This mapping information allows the diagnosis client to conduct diagnosis over the involved tunnel session, by invoking a separate Diagnostic query for the corresponding Tunnel Session and Tunnel Sender. Keep in mind, however, that multiple end-to-end sessions may all map to one pre-configured tunnel session that may have totally different parameter settings.

The tunnel object is defined in the RSVP Tunnel Specification [RSVPTUN].

4. Diagnostic Packet Forwarding Rules

4.1. DREQ Packet Forwarding

DREQ messages are forwarded hop-by-hop via unicast from the LAST-HOP address to the Sender address, as specified in the DIAGNOSTIC object. If an RSVP capable node, other than the LAST-HOP node, receives a DREQ message that contains no DIAG_RESPONSE objects and has a zero Fragment Offset, the node should forward the DREQ packet towards the LAST-HOP without doing any of the processing mentioned below. The reason is that

such conditions apply only for nodes downstream of the LAST-HOP where no information should be collected.

Processing begins when a DREQ message, DREQ_in, arrives at a node.

1. Create a new DIAG_RESPONSE object. Compute the IP hop count from the previous RSVP hop. This is done by subtracting the value of the TTL value in the IP header from Send_TTL in the RSVP common header. Save the result in the D-TTL field of the DIAG_RESPONSE object.

2. Set the DREQ Arrival Time and the Outgoing Interface Address in the DIAG_RESPONSE object. If this node is the LAST-HOP, then the Outgoing Interface Address field in the DIAG_RESPONSE object contains the following value depending on the session being diagnosed.

   * If the session in question is a unicast session, then the Outgoing Interface Address field contains the address of the interface LAST-HOP uses to send PATH messages and data to the receiver specified by the session address.

   * Otherwise, if it is a multicast session and there is at least one receiver for this session, LAST_HOP should use the address of one of local interfaces used to reach one of the receivers.

   * Otherwise Outgoing Interface Address should be zero.

3. Increment the RSVP-hop-count field in the DIAGNOSTIC message object by one.

4. If no PATH state exists for the specified session, set R-error = 0x01 (No PATH state) and goto step 7.

5. Set the rest of the fields in the DIAG_RESPONSE object. If DREQ_in contains a DIAG_SELECT object, the response object classes are those specified in the DIAG_SELECT; otherwise, they are SENDER_TSPEC, STYLE, and FLOWSPEC objects. If no reservation state exists for the specified RSVP session, the DIAG_RESPONSE object will contain no FLOWSPEC, FILTER_SPEC or STYLE object. If neither PATH nor reservation state exists for the specified RSVP session, then no response objects will be appended to the DIAG_RESPONSE object.

6. If RSVP-hop-count is less than Max-RSVP-hops and this node is not the sender, then the DREQ is eligible for forwarding; set the Path MTU to the min of the Path MTU and the MTU size of the incoming interface for the sender being diagnosed.

7. If the size of DREQ_in plus the size of the new DIAG_RESPONSE

object plus the size of an IP address (if a ROUTE object exists and R-error= 0) is larger than Path MTU, then the new diagnostic message will be too large to be forwarded or returned without fragmentation; set the "packet too big" (0x02) error bit in DIAG_RESPONSE and goto Step SD1 in Send_DREP (below).

8. If the "No PATH state" (0x01) error bit is set or if RSVP-hop-count is equal to Max-RSVP-hops or if this node is the sender, then the DREQ cannot be forwarded further; goto Step 10.

9. Forward the DREQ towards the sender, as follows. If a ROUTE object exists, append the "Incoming Interface Address" to the end of the ROUTE object and increment R-Pointer by one. Update the Next-Hop RSVP_HOP object, append the new DIAG_RESPONSE object to the list of DIAG_RESPONSE object, and update the message length field in the RSVP common header accordingly. Finally, recompute the checksum, forward DREQ_in to the next hop towards the sender, and return.

10. Turn the DREQ into a DREP and return to the requester, as follows. Append the DIAG_RESPONSE object to the end of DREQ_in and update the packet length. If a ROUTE object is present in the message, decrement the R-pointer and set target address to the last address in the ROUTE object, otherwise set target address to the requester address. Change the Type Field in the Common header from DREQ to DREP. Finally, recompute the checksum, send the DREP to the target address, and return. Note that the MF bit must be off in this case.

Send_DREP:

This sequence is entered if the DREQ message augmented with the new DIAG_RESPONSE object is too large to be forwarded towards the sender or, if it is not eligible for forwarding, too large to be returned as a DREP.

SD1. Make a copy of DREQ_in and change the message type field from DREQ to DREP. Trim all DIAG_RESPONSE objects from DREQ_in and adjust the Fragment Offset. The DREP message contains the DIAG_RESPONSE objects accumulated by prior nodes.

SD2. Send the DREP message towards the requester, as follows. If a ROUTE object is present in the DREP message, decrement the R-pointer and set target address to the last address in the ROUTE object, otherwise set target address to the requester address. Set the MF bit, recompute the checksum and send the DREP message back to the target address.

SD3. If the reduced size of DREQ_in plus the size of DIAG_RESPONSE plus the size of an IP address (if a ROUTE object exists) is smaller than or equal to Path MTU, then return to Step 8 of the main DREQ processing sequence above.

SD4. If a ROUTE object exists, replace the ROUTE object in DREQ_in with an empty ROUTE object and turn on the "ROUTE object too big" (0x04) error bit in the DIAG_RESPONSE. In either case, return to Step 8 of the main DREQ processing sequence above.

4.2. DREP Forwarding

When a ROUTE object is present, DREP messages are forwarded hop-by-hop towards the requester, by reversing the route as listed in the ROUTE object. Otherwise, DREP messages are sent directly to the original requester.

When a node receives a DREP message, it simply decreases R-pointer by one (address length), recomputes the checksum and forwards the message to the address pointed to by R-pointer in the route list. If a node, other than the LAST-HOP, receives a DREP packet where R-pointer is equal to zero, it must send it directly to the requester.

When the LAST-HOP node receives a DREP message, it sends the message to the requester.

4.3. MTU Selection and Adjustment

Because the DREQ message carries the allowed MTU size of previous hops that the DREP messages will later traverse, this unique feature allows easy semantic fragmentation as described above. Whenever the DREQ message approaches the size of Path MTU, it can be trimmed before being forwarded again.

When a requester sends a DREQ message, the Path MTU field in the DIAGNOSTIC object can be set to a configured default value. It is possible that the original Path MTU value is chosen larger than the actual MTU value along some portion of the path being traced. Therefore each intermediate RSVP node must check the MTU value when processing a DREQ message. If the specified MTU value is larger than the MTU of the incoming interface (that the DREQ message will be forwarded to), the node changes the MTU value in the header to the smaller value.

Whenever a DREQ message size becomes larger than the Path MTU value, an intermediate RSVP node makes a copy of the message, converts it to a DREP message to send back, and then trims off the partial results from

the DREQ message. If in this case also the DREQ cannot be forwarded upstream due to a large ROUTE object, the "ROUTE object too big" is set and the ROUTE object is trimmed. As a result of the ROUTE object trimming, DREP(s) will come hop-by-hop up to this node and will then immediately be forwarded to the requester address.

Even if the steps shown above are followed there are a few cases where fragmentation at the IP layer will happen. For example, non-RSVP hops with smaller MTUs may exist before LAST-HOP is reached, or if the response is sent directly back to requester (as opposed to hop by hop) the DREP may take a different route to the requester than the DREQ took from the requester. Another case is when there exists a link with MTU smaller than the minimum Path MTU value defined in Section 3.3.

4.4. Errors

If an error condition prevents a DREP message from being forwarded further, the message is simply dropped.

If an error condition, such as lack of PATH state, prevents a DREQ message from being forwarded further, the node must change the current message to DREP type and return it to the response address.

5. Problem Diagnosis by Using RSVP Diagnostic Facility

5.1. Across Firewalls

Firewalls may cause problems in diagnostic message forwarding. Let us look at two different cases.

First, let us assume that the querier resides on a receiving host of the session to be examined. In this case, firewalls should not prevent the forwarding of the diagnostic messages in a hop-by-hop manner, assuming that proper holes have been punched on the firewall to allow hop-by-hop forwarding of other RSVP messages. The querier may start by not including a ROUTE object, which can give a faster response delivery and reduced overhead at intermediate nodes. However if no response is received, the querier may resend the DREQ message with a ROUTE object, specifying that a hop-by-hop reply should be sent.

If the requester is a third party host and is separated from the LAST-HOP address by a firewall (either the requester is behind a firewall, or the LAST-HOP is a node behind a firewall, or both), at this time we do not know any other solution but to change the LAST-HOP to a node that is on the same side of the firewall as the requester.

5.2. Examination of RSVP Timers

One can easily collect information about the current timer value at each RSVP hop along the way. This will be very helpful in situations when the reservation state goes up and down frequently, to find out whether the state changes are due to improper setting of timer values, or K values (when across lossy links), or frequent routing changes.

5.3. Discovering Non-RSVP Clouds

The D-TTL field in each DIAG_RESPONSE object shows the number of routing hops between adjacent RSVP nodes. Therefore any value greater than one indicates a non-RSVP cloud in between. Together with the arrival timestamps (assuming NTP works), this value can also give some vague, though not necessarily accurate, indication of how big that cloud might be. One might also find out all the intermediate non-RSVP nodes by running either unicast or multicast trace route.

5.4. Discovering Reservation Merges

The flowspec value in a DIAG_RESPONSE object specifies the amount of resources being reserved for the data stream defined by the filter spec in the same data block. When this value of adjacent DIAG_RESPONSE objects differs, that is, a downstream node Rd has a smaller value than its immediate upstream node Ru, it indicates a merge of reservation with RSVP request(s) from other down stream interface(s) at Rd. Further, in case of SE style reservation, one can examine how the different SE scopes get merged at each hop.

In particular, if a receiver sends a DREQ message before sending its own reservation, it can discover (1) how many RSVP hops there are along the path between the specified sender and itself, (2) how many of the hops already have some reservation by other receivers, and (3) possibly a rough prediction of how its reservation request might get merged with other existing ones.

5.5. Error Diagnosis

In addition to examining the state of a working reservation, RSVP diagnostic messages are more likely to be invoked when things are not working correctly. For example, a receiver has reserved an adequate pipe for a specified incoming data stream, yet the observed delay or loss ratio is much higher than expected. In this case the receiver can use the diagnostic facility to examine the reservation state at each RSVP hop along the way to find out whether the RSVP state is set up

correctly, whether there is any blackhole along the way that caused RSVP message losses, or whether there are non-RSVP clouds, and where they are, that may have caused the performance problem.

5.6. Crossing "Legacy" RSVP Routers

Since this diagnosis facility was developed and added to RSVP after a number of RSVP implementations were in place, it is possible, or even likely, that when performing RSVP diagnosis, one may encounter one or more RSVP-capable nodes that do not understand diagnostic messages and drop them. When this happens, the invoking client will get no response from its requests.

One way to by-pass such "legacy" RSVP nodes is to perform RSVP diagnosis repeatedly, guided by information from traceroute, or mtrace in case of multicast. When an RSVP diagnostic query times out (see next section), one may first use traceroute to get the list of nodes along the path, and then gradually increase the value of Max-RSVP-hops field in the DREQ message, starting from a low value until one no longer receives a response. One can then try RSVP diagnosis again by starting with the first node (which is further upstream towards the sender) after the unresponding one.

There are two problem with the method mentioned above in the case of unicast sessions. Both problems are related to the fact that traceroute information provides the path from the requester to the sender. The first problem is that the LAST-HOP may not be on the path from the requester to the sender. In this case we can get information only from the portion of the path from the LAST-HOP to the sender which intersects with the path from the requester to the sender. If routers that are not on the intersection of the two paths don't have PATH state for the session being diagnosed then they will reply with R-error=0x01. The requester can overcome this problem by sending a DREQ to every router on the path (from itself to the sender) until it reaches the first router that belongs to the path from the sender to the LAST-HOP.

The second problem is that traceroute provides the path from the requester to the sender which, due to routing asymmetries, may be different than the path traffic from the sender to the LAST-HOP uses. There is (at least) one case where this asymmetry will cause the diagnosis to fail. We present this case below.

```
                                Downstream Path                Sender
                                __         __            __        __
   Receiver             +------|  |<------|  |<-- ...---|  |-----|  |
      __          __   /       |__|       |__|          |__|     |__|
     |  |--.....--|X |_/                   ^
     |__|        |__| \     Router B       |
                Black  \        __         |
                Hole    +----->|  |---->---+
                               |__| Upstream Path

                            Router A
```

Figure 2

Here the first hop upstream of the black hole is different on the upstream path and the downstream path. Traceroute will indicate router A as the previous hop (instead of router B which is the right one). Sending a DREQ to router A will result in A responding with R-error 0x01 (No PATH State). If the two paths converge again then the requester can use the solution proposed above to get any (partial) information from the rest of the path.

We don't have, for the moment, any complete solutions for the problematic scenarios described here.

6. Comments on Diagnostic Client Implementation.

Following the design principle that nodes in the network should not hold more than necessary state, RSVP nodes are responsible only for forwarding Diagnostic messages and filling DIAG_RESPONSE objects. Additional diagnostic functionality should be carried out by the diagnostic clients. Furthermore, if the diagnostic function is invoked from a third-party host, we should not require that host be running an RSVP daemon to perform the function. Below we sketch out the basic functions that a diagnostic client daemon should carry out.

1. Take input from the user about the session to be diagnosed, the last-hop and the sender address, the Max-RSVP-hops, and possibly the DIAG_SELECT list, create a DREQ message and send to the LAST-HOP RSVP node using raw IP message with protocol number 46 (RSVP). If the user specified that the response should be sent hop-by-hop include an empty ROUTE object to the DREQ message sent. Set the Path_MTU to the smaller of the user request and the MTU of the link through which the DREQ will be sent.

   The port of the UDP socket on which the Diagnostic Client is

listening for replies should be included in the Requester FILTER_SPEC object.

2. Set a retransmission timer, waiting for the reply (one or more DREP messages). Listen to the specified UDP port for responses from the LAST-HOP RSVP node.

   The LAST-HOP RSVP node, upon receiving DREP messages, sends them to the Diagnostic Client as UDP packets, using the port supplied in the Requester FILTER_SPEC object.

3. Upon receiving a DREP message to an outstanding diagnostic request, the client should clear the retransmission timer, check to see if the reply contains the complete result of the requested diagnosis. If so, it should pass the result up to the invoking entity immediately.

4. Reassemble DREP fragments. If the first reply to an outstanding diagnostic request contains only a fragment of the expected result, the client should set up a reassembly timer in a way similar to IP packet reassembly timer. If the timer goes off before all fragments arrive, the client should pass the partial result to the invoking entity.

5. Use retransmission and reassembly timers to gracefully handle packet losses and reply fragment scenarios.

   In the absence of response to the first diagnostic request, a client should retransmit the request a few times. If all the retransmissions also fail, the client should invoke traceroute or mtrace to obtain the list of hops along the path segment to be diagnosed, and then perform an iteration of diagnosis with increasing hop count as suggested in Section 5.6 in order to cross RSVP-capable but diagnosis-incapable nodes.

6. If all the above efforts fail, the client must notify the invoking entity.

7. Security Considerations

RSVP Diagnostics, as any other diagnostic tool, can be a security threat since it can reveal possibly sensitive RSVP state information to unwanted third parties.

We feel that the threat is minimal, since as explained in the Introduction Diagnostics messages produce no side-effects and therefore they

cannot change RSVP state in the nodes. In this respect RSVP Diagnostics is less a security threat than other diagnostic tools and protocols such as SNMP.

Furthermore, processing of Diagnostic messages can be disabled if it is felt that is a security threat.

8. Acknowledgments

The idea of developing a diagnostic facility for RSVP was first suggested by Mark Handley of UCL. Many thanks to Lee Breslau of Xerox PARC and John Krawczyk of Baynetworks for their valuable comments on the first draft of this memo. Lee Breslau, Bob Braden, and John Krawczyk contributed further comments after March 1996 IETF. Steven Berson provided valuable comments on various drafts of the memo. Tim Gleeson contributed an extensive list of editorial comments. We would also like to acknowledge Intel for providing a research grant as a partial support for this work. Subramaniam Vincent did most of this work while a graduate research assistant at the USC Information Sciences Institute (ISI).

9. References

[RSVP] Braden, R. Ed. et al, "Resource ReserVation Protocol -- Version 1 Functional Specification", RFC 2205, September 1997.

[RSVPTUN] A. Terzis, J. Krawczyk, J. Wroclawski, L. Zhang. "RSVP Operation Over IP Tunnels ", Internet Draft. draft-ietf-rsvp-tunnel-02.txt, February, 1999.

10. Authors' Addresses

Andreas Terzis
UCLA
4677 Boelter Hall
Los Angeles, CA 90095

Phone: 310-267-2190
Email: terzis@cs.ucla.edu

Bob Braden
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Phone: 310 822-1511
EMail: braden@isi.edu

Subramaniam Vincent
Cisco Systems
275, E Tasman Drive, MS SJC04/2/1
San Jose, CA 95134.
Phone: 408 525 3474
Email: svincent@cisco.com

Lixia Zhang
UCLA
4531G Boelter Hall
Los Angeles, CA 90095

Phone: 310-825-2695
EMail: lixia@cs.ucla.edu