Internet Engineering Task Force
INTERNET DRAFT

P. Pan/H. Schulzrinne/R. Guerin
IBM/Columbia University/IBM

21 November 1997

Staged Refresh Timers for RSVP
draft-pan-rsvp-timer-00.txt

Status of This Memo

Abstract

The current resource Reservation Protocol (RSVP) design has no reliability mechanism for the delivery of control messages. Instead, RSVP relies on periodic refresh between routers to maintain reservation states. This approach has several problems in a congested network. End systems send Path and Resv messages to set up RSVP connections. If the first Path or Resv message from an end system is accidentally lost in the network, a copy of the message will not be retransmitted until the end of a refresh interval, causing a delay of 30 seconds or more until a reservation is established. If a congested link causes a tear-down message (PathTear or ResvTear) to be dropped, the corresponding reservation will not be removed from the routers until the RSVP cleanup timer expires.

We present an RSVP enhancement called staged refresh timers to support fast and reliable message delivery that ensures hop-by-hop delivery of control messages without violating the soft-state design. The enhancement is backwards-compatible and can be easily added to current implementations. The new approach can speed up the delivery of trigger messages while reducing the amount of refresh messages. The approach is also applicable to other soft-state protocols.

The performance benefits of this approach are explored in [PS97].

Contents

1. Introduction

The Reservation Protocol (RSVP) [ZDE+93, BZB+97] has been designed to exchange resource reservation information among routers in an internet. One of its advantages is that it relies on soft state to maintain reservation state in each router: Reservations will disappear by themselves if they are not refreshed periodically. This avoids orphan reservations and allows reservations to adapt quickly to routing changes, without involvement of the end systems. End systems send explicit tear-down messages to speed up the removal of reservations when routes change or the application exits.

RSVP sends its control messages as IP datagrams with no reliability guarantee. It relies on the periodic refresh messages from hosts and routers to handle the occasional loss of a Path or Resv message. Each RSVP host or router maintains a cleanup timer. A state is deleted if no refresh messages arrive before the expiration of a cleanup timeout interval.

Packet losses in the current Internet can be frequent, unfortunately. In today's Internet multicast backbone (Mbone), the packet loss rate [YKT96] is approximately 1-2% on average, and can occasionally reach 20% or more on congested links. The existing RSVP message delivery mechanism will not work well in such an environment. For example, when a user tries to make a reservation over the network, if the first reservation request (Resv message) is lost due to congestion, it will not be retransmitted over the congested link until the next refresh cycle arrives. The default refresh interval is 30 seconds.

Thus, the first few seconds of, say, a multimedia flow may experience degraded quality of service as packets are carried on a best-effort basis rather than as a reserved flow. Unfortunately, packet loss is more likely to delay reservations just when needed most, i.e., when packet loss rates for best-effort service are high.

RSVP soft states are managed hop-by-hop, i.e., no network entities other than the node that sent the original refresh message can retransmit a refresh message. Thus, a user cannot accelerate the reservation process by retransmitting RSVP messages.

RSVP also does not retransmit tear-down messages. If, for example, a user tries to remove a reservation, and the message (ResvTear) is lost, the reservation will remain in place until it times out, by default after 90 seconds. If holding a reservation incurs costs, the user will have to pay for the extra time that has been spent waiting for the reservation time-out. Also, network resources are used inefficiently. Network providers will have to account for this uncertainty in their billing policies.

In this document, we propose a simple RSVP extension that provides a mechanism to deliver RSVP messages faster and more reliably, that is backward compatible with the existing implementations, and that reduces the number of refreshes among routers, contributing to protocol scalability.

2. Terminology

2.1. Sending and Receiving Nodes

A sending node is a router or host that generates RSVP messages. A receiving node is defined as the RSVP router or host that is one RSVP hop away from a sending node. In a shared-media or non-broadcast multiple access (NBMA) network such as an ATM subnet, a sending node may have multiple receiving nodes. In some cases, not all routers between sending and receiving nodes implement RSVP. We refer to these networks as non-RSVP clouds.

2.2. Trigger and Refresh Message

In RSVP, control traffic can be categorized into two types: trigger and refresh messages. Trigger messages are generated by an RSVP host or a router due to state changes. Such state changes include the initiation of a new state, a route change that altered the reservation paths, or a reservation modification by a downstream router. Path, Resv, PathTear and ResvTear serve as RSVP trigger messages.

Refresh messages, on the other hand, contain replicated state information generated by a router to maintain state. As indicated in the introduction, RSVP periodically refreshes state for robustness. For instance, if the RSVP daemon on a router crashes and resets, it loses all RSVP state information. However, since its neighbor routers send copies of RSVP state information periodically, the router can recover the lost states within one refresh interval. A refresh message can be either a Path or Resv message.

The RSVP routing interface [Zap96, GSE97] can detect state changes, so that refresh messages are not needed to update router reservation states. If the RSVP daemon is reasonably reliable, refresh messages are more of a safety mechanism than actually used for network operation and can thus be sent very infrequently, in the range of hours instead of 30 seconds. This greatly reduces the traffic and processing impact of RSVP messages and makes RSVP signaling at least as efficient as circuit-switched setup protocols. However, this requires that trigger messages are delivered reliably.

3. Protocol Mechanisms

3.1. Outline of Operation

We propose the following feedback mechanism for RSVP trigger message delivery: When sending an RSVP trigger message, a node inserts a new echo-request flag into the RSVP common header of the message. Upon reception, a receiving node acknowledges the arrival of the message by sending back an echo-reply. When the sending node receives this echo-reply for a Path or Resv message, it will automatically scale back the refresh rate for these messages for the flow. If the trigger message was a flow tear-down, no more tear-down messages are sent, just as in the current RSVP specification. Until the echo reply is received, the sending node will retransmit the trigger message. The interval between retransmissions is governed by a staged refresh timer. The staged refresh timer starts at a small interval which increases exponentially until it reaches a threshold. From that point on, the sending node will use a fixed timer to refresh Path and Resv messages and stop re-transmitting tear-down messages. This mechanism is designed so that the message load is only slightly larger than in the current specification even if a node does not support this staged refresh timer.

3.2. Time Parameters

The new extension makes the use of the following time parameters. All parameters should be modifiable per interface.

Fast refresh interval Rf:

Rf is the initial retransmission interval for trigger messages. After sending the message for the first time, the sending node will schedule a retransmission after Rf seconds. The value of Rf could be as small as the round trip time (RTT) between a sending and a receiving node, if known. Unless a node knows that all receiving nodes support echo-replies, a slightly larger value of, for example, 3 seconds is suggested.

Slow refresh interval Rs:

The sending node retransmits with this interval after it has determined that the receiving nodes support the RSVP echo-reply. To reduce the number of unnecessary refreshes in a stable network, Rs can be set to a large value. The value of Rs can be set for each egress interface. Throughout the remainder of the document we assume a value of 15 minutes for Rs.

Fixed refresh interval R:

A node retransmits the trigger message with the interval Rf*(1 + Delta)**i until the refresh interval reaches the fixed refresh interval R or an echo reply has been received. If no reply has been received, the node continues to retransmit refreshes every R seconds. We choose a value for R of 30 seconds, the same value as the refresh interval in the current RSVP specification.

Increment value Delta:

Delta governs the speed with which the sender increases the refresh interval. The ratio of two successive refresh intervals is (1 + Delta). We arbitrarily set Delta to 0.30, which is also the the same value as the Slew.Max parameter that has been defined in RSVP to increase the retransmission and timeout interval for long-lived flows using local repairs.

3.3. Staged Refresh

After a sending node transmits a trigger message, it will immediately schedule a retransmission after Rf seconds. If it receives echo-replies, the sending node will change the refresh interval to Rs. Otherwise, it will retransmit the message after (1 + Delta)*Rf seconds. The staged retransmission will continue until either echo-replies are received, or the refresh interval has been increased to R.

The implementation of staged refresh is simple. A sending node can use the following algorithm when the RSVP refresh timer for state (flow) k has expired:

```
 if (Rk < R)  {
     Rk = Rk * (1 + Delta);
     send out a refresh message;
     wake up in state k after Rk seconds;
     exit;
 }
 else  {    /* no reply from receivers for too long: */
     Rk = R;
     if (the state k is for a tear-down message) {
         clean up state k;
         exit;
     }
     else  {
```

```
            send out a refresh message;
            wake up state k after Rk seconds;
            exit;
        }
    }
```

Asynchronously, when a sending node receives echo-replies from the receiving nodes, it will change the refresh interval Rk to Rs for state k.

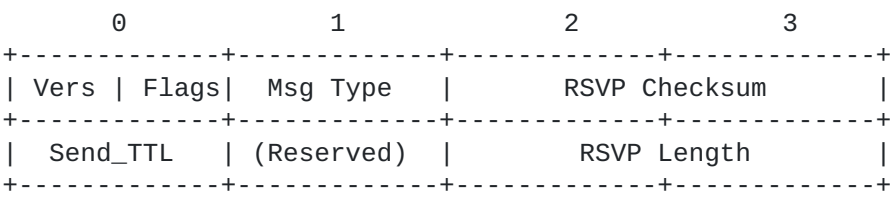
4. Protocol Extension

The proposed mechanism requires several minor modifications to the current version of RSVP: a new bit is defined in the flag field of the RSVP common header, and four new message types are created for echo-reply. The echo reply messages are simple copies of the message to be confirmed, with the message type changed. While Path messages are generated end-to-end, Path echo-replies are hop-by-hop, using the previous hop (PHOP) field from the message.

4.1. Common Header

For Path, Resv, PathTear and ResvTear messages, there is an additional echo-request flag in the RSVP common header. Four additional new messages have also been defined to support feedback.

The format of the RSVP common-header is:



Flags: 4 bits

0x01: echo-request flag.

Msg Type: 8 bits

8 = PathAck
9 = PathTearAck
10 = ResvAck
11 = ResvTearAck

4.2. Echo-Reply Messages

4.2.1. PathAck Messages

The format of a PathAck message is as follows:

<PathAck Message> ::= <Common Header> [<INTEGRITY>]

<SESSION> <RSVP_HOP>

[<sender descriptor>]

<sender descriptor> ::= <SENDER_TEMPLATE> [<SENDER_TSPEC>]

[<ADSPEC>]

The RSVP_HOP object of each PathAck message contains the IP address of the interface through which the Path message was received and the LIH (Logical Interface Handle) which was carried in the Path message.

The destination address in IP header is the address stored in the RSVP_HOP object of the original Path message.

4.2.2. PathTearAck Messages

The format of a PathTearAck message is as follows:

```
<PathTearAck Message> ::= <Common Header> [ <INTEGRITY> ]
                          <SESSION> <RSVP_HOP>
                          [ <sender descriptor> ]
<sender descriptor> ::= <SENDER_TEMPLATE> [ <SENDER_TSPEC> ]
                          [ <ADSPEC> ]
```

The RSVP_HOP object of each PathTearAck message contains the IP address of the interface through which the PathTear message was received and the LIH of which was carried in the PathTear message.

The destination address in IP header is the address stored in the RSVP_HOP object of the original PathTear message.

4.2.3. ResvAck Messages

```
<ResvAck Message> ::= <Common Header> [<INTEGRITY>]
                      <SESSION> <RSVP_HOP>
                      [ <SCOPE> ] <STYLE>
                      <flow descriptor list>
<flow descriptor list> ::=  (see RSVP specification, RFC2205)
```

The RSVP_HOP (i.e., the PHOP) object contains the IP address of the interface through which the Resv message was received. It also carries the corresponding LIH.

4.2.4. ResvTearAck Messages

```
<ResvTearAck Message> ::= <Common Header> [<INTEGRITY>]
                          <SESSION> <RSVP_HOP>
```

```
                                [ <SCOPE> ] <STYLE>

                                <flow descriptor list>

          <flow descriptor list> ::=  (see RSVP specification, RFC2205)
```

The RSVP_HOP (i.e., the PHOP) object contains the IP address of the interface through which the ResvTear message was received. It also carries the corresponding LIH.

5. Special Considerations

5.1. Backward Compatibility

Backward compatibility is one of the main objectives in our design. One cannot assume that both sending and receiving nodes on a link will support the extension simultaneously.

In the current RSVP specification, sending nodes refresh the soft states with fixed timers. In our design, sending nodes rely on echo request/reply mechanism to ``learn'' about the status of receiving nodes. If a sending node does not receive echo replies from the receiving nodes after several tries, it will assume the receiving nodes do not support the new extension, and switch its refresh interval to a fixed value. The RSVP operation is not affected at the receiving nodes.

5.2. Computing Cleanup Timeout Values

Each RSVP Path and Resv message carries a refresh interval in its TIME_VALUES object. Receiver nodes use the refresh interval to compute the cleanup timeout interval that governs the lifetime of reservation state that has not been refreshed. Generally, the cleanup timeout interval is a small multiple of refresh interval.

In the staged refresh design, a sending node initially places the slow refresh timer, Rs, in the Path or Resv message. For the receiving nodes that do not support the new extension, the sending node will insert R in the refresh messages after the actual refresh interval has been increased to R. If the receiving nodes do support the new extension, they will set the cleanup timeout interval based on Rs.

5.3. Handling of Tear-Down Messages

RSVP uses PathTear and ResvTear messages to tear-down path and reservation states, respectively. According to the current specification, sending nodes only generate one tear-message per flow. If the message is accidentally dropped along the way, the reserved resource will not be released until the cleanup timer expires. However, receiving duplicate tear-down messages at a receiving node should not impact the operation of RSVP in a proper implementation.

In our RSVP extension, we have altered the processing rules for tear-down messages at the sending node. Instead of deleting the state after a tear-down message is sent, a sending node will release all resource allocated to the state, and mark the state as closing. The state information is saved for message retransmission. The entire state information will be removed when echo-replies are received, or when the sending node realizes that the receiving nodes do not support the extension.

5.4. Operation in an NBMA Environment

For a multicast RSVP session in a non-broadcast multiple access (NBMA) network (such as ATM), a sending node may not know the total number of receiving nodes for a Path or PathTear message at an egress interface. Therefore, a sending node cannot simply switch to the longer refresh timer Rs based on having received echo-replies.

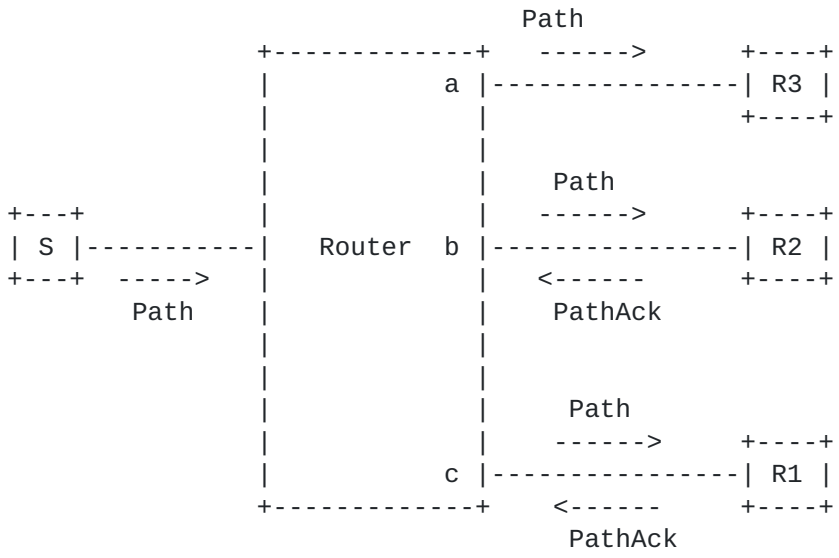


Figure-1: Path/PathAck in an NBMA network

For example, as shown in Figure-1, if the receiving node R3 does not support the new RSVP extension, the sending node S should not change to the longer refresh interval Rs, even though it has received echo-replies from R1 and R2.

In this case, a sending node has two alternatives:

- It can query a local database such as the ARP or MARS server to find out the exact number of the next-hop receivers. It then switches to a longer refresh interval after receiving echo-replies from all receiving nodes.

- Since Path messages are mainly used for traffic advertisement purposes, the sending node may not need to use staged refresh timers for Path messages. In an NBMA network, the staged refresh time mechanism would only make sense for the message delivery of Resv, ResvTear and PathTear messages.

In case of PathTear message, a sending node always knows all the receiving nodes that have made reservations. The following rules can be used:

- A sending node stops re-transmitting PathTear messages once it receives echo-replies from all its known next-hop receivers at an egress interface.

- Otherwise, the sending node generates PathTear messages using staged refresh timer until the refresh interval is increased to the fixed refresh rate R. Then it stops re-transmitting.

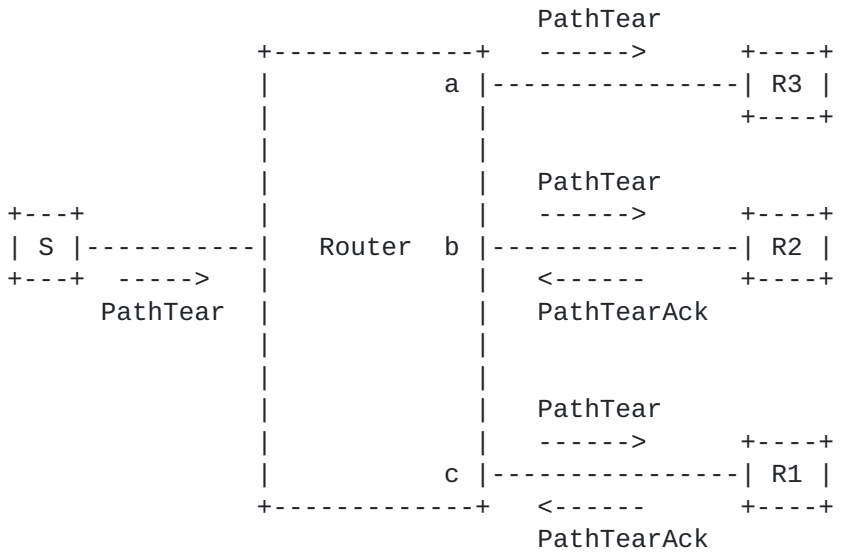


Figure-2: PathTear/PathTearAck in an NBMA network

An example is shown in Figure-2. R1, R2 and R3 are the receiving nodes to S. Initially, the sender S had the reservation state information for receiving node R1 and R2. Since R3 did not make any reservation, S would not know the existence of R3 from its RSVP database. After sending the first PathTear message, S will retransmit the message until it has received echo replies from R1 and R2. After which, S stops generating PathTear messages.

6. Discussion

We believe that RSVP message delivery mechanism requires some degree of reliability guarantee to make RSVP useful for individual applications rather than reserving ``pipes''. One way of improving reliability is to grant some minimal bandwidth for RSVP messages to protect them from congestion losses, as suggested in the RSVP

specification [BZB+97]. However, this may require additional functionality at both sending and receiving nodes and does not help if RSVP messages have to traverse non-RSVP clouds. It is also not clear how this can be achieved in a backward-compatible manner.

In this paper, we presented a mechanism called staged refresh timers that enhances the current RSVP message delivery and is completely backward compatible. Staged refresh timers are easy to add to RSVP router and host implementations and save both processing and bandwidth overhead.

The staged refresh timer mechanism is an example of state management that falls somewhere between ``classical'' handshake-based reliability as found in ATM signaling, for example, and purely timer-based soft-state protocols such as the original RSVP proposal [ZDE+93], delta-t [Wat89] or IGMPv1 [Dee89]. An approach similar to staged refresh is also being used by the Session Initiation Protocol [HSS97] to confirm state establishment. Generally speaking, experience has shown that state management is greatly simplified by requiring only one message (in each direction) to establish state, rather than going through several intermedia states. State establishment messages should be idempotent and should contain a globally (spatially and temporally) unique state label, so that retransmissions of the same message can be ignored.

Since only neighboring routers are involved in the reliability mechanism described here, these routers can easily estimate round-trip times, thus further tightening the retransmission interval, if desired.

While staged refresh timers improve scalability, RSVP remains a rather complex protocol. Alternative approaches to reserve reservation [CW97, PS98] may offer better scaling properties.

A. Appendix: Processing Rules

We outline a set of algorithms to illustrate how the new scheme works. The notations and definitions such as PSB and RSB are defined in [BZ97]. However, this should not be considered as the actual implementation requirements.

A.1. Modification to the Existing Rules

A.1.1. PATH MESSAGE ARRIVES:

Two additional steps that are required during Path message processing:

1. reply PathAck message back to a sending node;

2. use staged refresh timer to send Path if the PSB is new or changed.

Reference Algorithm:

Assume the message arrives on interface InIf.

After message sanity checks:

o Search for a path state block (PSB) whose (session, sender_template) pair matches the corresponding objects in the message, and whose ingress interface matches InIf.

o If there is no matching PSB, then:

    1. Create a new PSB.

    2. Update all relevant information to the PSB.

o Otherwise (there is a matching PSB):

    - Update all relevant information to the PSB.

o If the echo-request flag is set in the common-header, and the echo-reply option is enabled for the ingress interface InIf:

    - Execute the PATH ACK sequence (below) for the PHOP in PSB.

o Continue the Path message processing.

o If the PSB is new or modified, then:

    1. Update the PSB's refresh timer to fast refresh interval Rf.

    2. Execute the PATH REFRESH sequence.

A.1.2. PATHTEAR MESSAGE ARRIVES:

The router can reply a PathTearAck message to the sending node, release allocated resource and relay tear-down message downstream with staged refresh timer.

During the process of PathTear message, unlike what's in the current RSVP specification, where a state is deleted as a result, the state is marked as PathTear_Pending in the new scheme. It will be deleted by either the arrival of PathTearAck messages, or clean-up timer expiration.

Reference Algorithm:

Assume the message arrives on interface InIf.

After message sanity checks:

o Search for a PSB whose (Session, Sender_Template) pair matches the corresponding objects in the message, and the ingress interface matches to InIf.

o If there is no matching PSB:

  - Drop the message and return.

o If the echo-request flag is set in the common-header, and the echo-reply option is enabled for the ingress interface,

  - Execute the PATH ACK sequence (below) for the PHOP in PSB.

o Forward a copy of the PathTear message to each egress interface listed in the PSB.

o If the PSB is marked as Pathtear_Pending, (that is, a repeated PathTear message from upstream)

  - Drop the message and return.

o Otherwise, (i.e., the Pathtear_Pending flag in the PSB is off):

  1. Insert a Pathtear_Pending flag into the PSB.

  2. Release all resource associated to the PSB.

  3. Update the PSB's refresh timer to fast refresh interval Rf.

o Drop the message and return.

A.1.3. RESV MESSAGE ARRIVES:

Similar to the modified Path processing rule, the new extension requires a router to reply an acknowledgment message and schedule refreshes with staged refresh timer.

Reference Algorithm:

After message sanity checks:

o Find all reservation state blocks (RSB's) that are corresponding to the flows defined in the message.

- Execute RESV ACK sequence (below) to the NHOP in the message.

o Continue the Resv message processing.

o For all the RSB's that are new or modified,

1. Update their refresh interval to fast refresh interval Rf.

2. Execute RESV REFRESH sequence.

A.1.4. RESVTEAR MESSAGE ARRIVES:

The router will send an echo-reply to the sending node, and schedule refreshes with the staged refresh timer.

During the process of the message, resource that is associated with the state should be released and marked as pending. The state information is finally removed during the processing of ResvTearAck message or timer expiration.

Reference Algorithm:

After message sanity checks:

o Find all PSB's that are corresponding to the flows defined in

the message.

- Execute RESV ACK sequence (below) to the NHOP in the message.

o Continue the ResvTear message processing.

o For all the RSB's that are described the ResvTear message:

1. If a RSB is not marked with Resvtear_Pending flag:

- remove the reserved resource indicated in the RSB.

- mark the RSB to be Resvtear_Pending.

- mark the RSB's refresh interval to fast interval Rf.

2. Execute RESV REFRESH sequence to forward ResvTear messages upstream.

o Drop the message and return.

A.2. Processing Rules for New Messages

A.2.1. PATHACK MESSAGE ARRIVES

From SESSION, SENDER_TEMPLATE and egress interface information, a router finds the corresponding state and changes the refresh rate to a slower one.

Reference Algorithm:

Assume the message arrives on interface OutIf.

o Search for a path state block (PSB) whose (session, sender_template) pair matches the corresponding objects in the message, and whose egress interface matches OutIf.

o If there is no matching PSB, then:

- Drop the message and return.

o Otherwise (there is a matching)

- Set the refresh timer of the PSB to slow refresh interval Rs.

o Drop the message and return.

A.2.2. PATHTEARACK MESSAGE ARRIVES

From SESSION, SENDER_TEMPLATE and egress interface, a router finds the corresponding state. However the state can not be removed until the router has received acknowledgments from all known next-hop routers of the RSVP flow.

Reference Algorithm:

Assume the message arrives on interface OutIf.

o Search for a path state block (PSB) whose (session, sender_template) pair matches the corresponding objects in the message, and whose egress interface matches OutIf.

o If there is no matching PSB, then:

  - drop the message and return.

o Find the RSB that matches the PSB. (The resource associated with the RSB should has been released during PathTear processing time.)

  - remove the RSB.

o Search for all RSB that matches this PSB.

  - If no more RSB could be found, then:

    - remove the PSB and return.

  - Otherwise, (there may still be reserved flows downstream):

    - return.

A.2.3. RESVACK MESSAGE ARRIVES

Find all the states that are described in the message, and switches their refresh timer to a slower one.

Reference Algorithm:

After message sanity checks:

- o Find all the RSB's that are corresponding to the flows defined in the message.
  - Set the refresh timer of the RSB's to slow refresh interval Rs.
- o Drop the message and return.

A.2.4. RESVTEARACK MESSAGE ARRIVES

Find the corresponding states, and remove them.

Reference Algorithm:

After message sanity checks:

- o Find all RSB's that are corresponding to the flows defined in the message.
  - Delete the RSB's.
- o Drop the message and return.

A.2.5. PATH ACK

This sequence sends a PathAck or a PathTearAck towards a particular previous hop. It is invoked from either PATH MESSAGE ARRIVES, or PATHTEAR MESSAGE ARRIVES sequence.

- o Copy SESSION object and sender descriptor from the received Path (or PathTear) message into the PathAck (or PathTearAck)

message being built.

- Insert into its RSVP_HOP object the ingress interface address and the LIH for the interface.
- Insert the PHOP address (from the Path/PathTear message) as the destination address into the IP header being built.
- Update the RSVP common-header and IP header.
- Send the message out.

A.2.6. RESV ACK

This sequence sends a ResvAck or a ResvTearAck towards a particular next hop. It is invoked from either RESV MESSAGE ARRIVES, or RESVTEAR MESSAGE ARRIVES sequence.

- Copy SESSION, STYLE, SCOPE (if WF style), and the flow descriptor list from the received Resv (or ResvTear) message into the Resv (or ResvTearAck) message being built.
- Insert into its RSVP_HOP objet the egress interface address and the LIH for the interface.
- Insert the NHOP address (from the Resv/ResvTear message) as the destination address into the IP header being built.
- Update the RSVP common-header and IP header.
- Send the message out.

References

[BZ97] R. Braden and L. Zhang. Resource reservation protocol (rsvp) -- version 1 message processing rules. RFC 2209, Internet Engineering Task Force, September 1997.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) -- version 1

functional specification. , Internet Engineering Task Force, September 1997.

[CW97] D. Clark and J. Wroclawski. An approach to service allocation in the internet. Internet Draft, Internet Engineering Task Force, August 1997. Work in progress.

[Dee89] S. Deering. Host extensions for IP multicasting. STD 5, RFC 1112, Internet Engineering Task Force, August 1989.

[GSE97] R. Guerin, Kamat S., and Rosen E. Extended rsvp-routing interface. Internet Draft, Internet Engineering Task Force, July 1997. Work in progress.

[HSS97] Mark Handley, Henning Schulzrinne, and Eve Schooler. SIP: Session initiation protocol. Internet Draft, Internet Engineering Task Force, July 1997. Work in progress.

[PS97] Ping Pan and Henning Schulzrinne. Staged refresh timers for RSVP. In Global Internet'97, Phoenix, Arizona, November 1997. also IBM Research Technical Report TC20966.

[PS98] Ping P. Pan and Henning Schulzrinne. YESSIR: A simple reservation mechanism for the Internet. In IBM Research Technical Report TC20967, 1998.

[Wat89] Richard W. Watson. The Delta-t transport protocol: Features and experience. In Harry Rudin and Robin Williamson, editors, First IFIP WG6.1/WG6.4 International Workshop on Protocols for High-Speed Networks, pages 3--17, Z rich, Switzerland, May 1989.

[YKT96] Maya Yajnik, Jim Kurose, and Don Towsley. Packet loss correlation in the MBone multicast network. In Proceedings of Global Internet, London, England, November 1996.

[Zap96] Daniel Zappala. RSRR: a routing interface for RSVP. Internet Draft (expired), Internet Engineering Task Force, November 1996. Work in progress.

[ZDE+93] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: a new resource ReSerVation protocol. IEEE Network, 7(5):8--18, September 1993.

Authors' Address

Ping Pan
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
USA
Phone: +1 914 784-6579
Email: pan@watson.ibm.com

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027
USA
electronic mail: schulzrinne@cs.columbia.edu

Roch Guerin
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
USA
Phone: +1 914 784-7038
Email: guerin@watson.ibm.com